


```

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SSSSSS
LL          II         SSSSSS
LL          II         SS
LL          II         SS
LL          II         SS
LL          II         SS
LLLLLLLLLLL IIIIIIII   SSSSSSSS
LLLLLLLLLLL IIIIIIII   SSSSSSSS

```

(1)	62	DECLARATIONS
(1)	200	REGISTER DEFINITIONS
(1)	332	SPEED CONVERSION TABLES
(1)	390	CONTROLLER INITIALIZATION
(1)	440	CONTROLLER INITIALIZATION
(1)	516	UNIT INITIALIZATION
(1)	668	MAINTENANCE ROUTINES
(1)	742	OUTPUT MODEM CONTROL
(1)	783	RECEIVER INTERRUPT SERVICE
(1)	900	START I/O ROUTINE
(1)	994	PORT DMA ROUTINES
(1)	1290	PORT ROUTINES STOP, RESUME, XON, XOFF
(1)	1437	OUTPUT INTERRUPT SERVICE
(1)	1653	SET SPEED, PARITY PARAMETERS


```
0000 1      .TITLE YFDRIVER - Port Driver for DHU/DHV
0000 2      .IDENT 'V04-000'
0000 3
0000 4      *****
0000 5      *
0000 6      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      *  ALL RIGHTS RESERVED.
0000 9      *
0000 10     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     *  TRANSFERRED.
0000 16     *
0000 17     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     *  CORPORATION.
0000 20     *
0000 21     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     *
0000 24     *
0000 25     *****
0000 26
0000 27     ++
0000 28     FACILITY:
0000 29
0000 30     VAX/VMS TERMINAL DRIVER
0000 31
0000 32     ABSTRACT:
0000 33
0000 34     DHU/DHV ASYNC PORT DRIVER
0000 35
0000 36     AUTHOR:
0000 37
0000 38     RICK SPITZ/ANDREW PALKA
0000 39
0000 40     Revision history:
0000 41
0000 42     V03-004 LMP0275      L. Mark Pilant,      12-Jul-1984 21:05
0000 43     Initialize the ACL info in the ORB to be a null descriptor
0000 44     list rather than an empty queue. This avoids the overhead
0000 45     of locking and unlocking the ACL mutex, only to find out
0000 46     that the ACL was empty.
0000 47
0000 48     V03-003 EMD0097      Ellen M. Dusseault    30-Apr-1984
0000 49     Fix a few bugs - device timeout problem and test the
0000 50     abort flag first to see if it is necessary to clear it.
0000 51     Also add DEV$M_NNM characteristic to DEVCHAR2 so that
0000 52     these devices will have the prefix "node$".
0000 53
0000 54     V03-002 TMH0002      Tim Halvorsen          14-Apr-1984
0000 55     Fix references to UCBS$L_OWNUIC and UCBS$W_VPROT to use ORB.
0000 56
0000 57     V03-001 EMD0070      Ellen M. Dusseault      30-MAR-1984
```


YFDRIIVER
V04-000

- Port Driver for DHU/DHV

M 5

16-SEP-1984 02:26:48 VAX/VMS Macro V04-00
5-SEP-1984 04:17:43 [TTDRVR.SRC]YFDRIIVER.MAR;1

Page 2
(1)

0000 58 :
0000 59 :
0000 60 :--

Modify to make code more efficient.

YFDI
V04-

```
0000 62 .SBTTL DECLARATIONS
0000 63
0000 64 :
0000 65 : EXTERNAL DEFINITIONS:
0000 66 :
0000 67 $PRDEF
0000 68 $ACBDEF : DEFINE ACB
0000 69 $CRBDEF : DEFINE CRB
0000 70 $DCDEF : DEFINE ADAPTER TYPES
0000 71 $DDBDEF : DEFINE DDB
0000 72 $DEVDEF : DEFINE DEVICE TYPES
0000 73 $DYNDEF : DEFINE DYNAMIC STRUCTURE TYPES
0000 74 $IDBDEF : DEFINE IDB OFFSETS
0000 75 $IODEF : DEFINE I/O FUNCTION CODES
0000 76 $IRPDEF : DEFINE IRP
0000 77 $ORBDEF : DEFINE OBJECT RIGHTS BLOCK
0000 78 $SSDEF : DEFINE SYSTEM STATUS CODES
0000 79 $TTYDEF : DEFINE TERMINAL DRIVER SYMBOLS
0000 80 $TTDEF : DEFINE TERMINAL TYPES
0000 81 $TT2DEF : DEFINE EXTENDED CHARACTERISTICS
0000 82 $TQDEF : DEFINE TIMER QUEUE OFFSETS
0000 83 $UCBDEF : DEFINE UCB
0000 84 $UBADEF : DEFINE UBA
0000 85 $VECDDEF : DEFINE VECTOR FOR CRB
0000 86 $TTYMACS : DEFINE TERMINAL DRIVER MACROS
0000 87 $TTYDEFS : DEFINE TERMINAL DRIVER SYMBOLS
0000 88 $TTYMODEM : DEFINE MODEM DEFINITIONS
0000 89
0000 90
0000 91
0000 92 :
0000 93 : LOCAL STORAGE
0000 94 :
0000 95 .PSECT $$$105_PROLOGUE
0000 96
0000 97 :
0000 98 : Driver prologue table:
0000 99 :
0000 100
0000 101 :
0000 102 : Note. The DPT says that this driver has a UBA adapter type.
0000 103 : In fact it will work with a Q-bus VAX, either mapped or unmapped
0000 104 :
0000 105 YFSDPT::
0000 106 DPTAB - : DRIVER START
0000 107 END=YFSEND,- : DRIVER PROLOGUE TABLE
0000 108 UCBSIZE=UCB$C TT LENGTH+4,- : End and offset to INIT's vectors
0000 109 FLAGS=DPT$M_NOUNLOAD,- : NO UNLOAD ALLOWED
0000 110 ADAPTER=UBA,- : ADAPTER TYPE
0000 111 MAXUNITS=16,- : Number of units to create
0000 112 DEFUNITS=16,- : Number of units to create
0000 113 NAME=YFDRIVER,- : NAME OF DRIVER
0000 114 VECTOR=PORT VECTOR,- : PORT DRIVER VECTOR TABLE
0000 115 DELIVER=YF$DELIVER : Unit delivery routine
0038 116 DPT_STORE INIT
0038 117 DPT_STORE UCB,UCB$B_FIPL,B,8 : FORK IPL
003C 118 DPT_STORE UCB,UCB$L_DEVCHAR,L,<- : CHARACTERISTICS
```



```
003C 119 DEV$M_REC!- ;
003C 120 DEV$M_AVL!- ;
003C 121 DEV$M_IDV!- ;
003C 122 DEV$M_ODV!- ;
003C 123 DEV$M_TRM!- ;
003C 124 DEV$M_CCL> ;
0043 125 DPT_STORE UCB,UCB$S_DEVCHAR2,L,- ; DEVICE CHARACTERISTICS
0043 126 <DEV$M_NNM> ; PREFIX WITH 'NODE$'
004A 127 DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$ TERM;
004E 128 DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$ UNKNOWN ; TYPE
0052 129 DPT_STORE UCB,UCB$W_TT_DESIZE,AW,TTY$GW_DEFBUF ; BUFFER SIZE
0059 130 DPT_STORE UCB,UCB$S_TT_DECHAR,AL,TTY$GL_DEFCHAR ; DEFAULT CHARACTERS
0060 131 DPT_STORE UCB,UCB$S_TT_DECHA1,AL,TTY$GL_DEFCHAR2; DEFAULT CHARACTERS
0067 132 DPT_STORE UCB,UCB$W_TT_DESPEE,AB,TTY$GB_DEFSPEED; DEFAULT SPEED
006E 133 DPT_STORE UCB,UCB$W_TT_DESPEE+1,AB,TTY$GB_RSPEED; DEFAULT RSPEED
0075 134 DPT_STORE UCB,UCB$B_TT_DEPARI,AB,TTY$GB_PARITY ; DEFAULT PARITY
007C 135 DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$ UNKNOWN ; TYPE
0080 136 DPT_STORE UCB,UCB$W_DEVBUFSIZ,AW,TTY$GW_DEFBUF ; BUFFER SIZE
0087 137 DPT_STORE UCB,UCB$S_DEVDEPEND,AL,TTY$GL_DEFCHAR ; DEFAULT CHARACTERS
008E 138 DPT_STORE UCB,UCB$S_DEVDEPND2,AL,TTY$GL_DEFCHAR2; DEFAULT CHARACTERS
0095 139 DPT_STORE UCB,UCB$W_TT_SPEED,AB,TTY$GB_DEFSPEED ; DEFAULT SPEED
009C 140 DPT_STORE UCB,UCB$W_TT_SPEED+1,AB,TTY$GB_RSPEED ; DEFAULT RSPEED
00A3 141 DPT_STORE UCB,UCB$B_DIPL,B,21 ; DEVICE IPL
00A7 142 DPT_STORE UCB,UCB$S_TT_WFLINK,L,0 ; Zero write queue.
00AE 143 DPT_STORE UCB,UCB$S_TT_WBLINK,L,0 ; Zero write queue.
00B5 144 DPT_STORE UCB,UCB$S_TT_RTIMOU,L,0 ; Zero read timed out disp.
00BC 145 DPT_STORE ORB,ORB$B_FLAGS,B,- ; Protection block flags
00BC 146 <ORB$M_PROT 16> ; SOGW protection word
00C0 147 DPT_STORE ORB,ORB$W_PROT,AW,TTY$GW_PROT ; Default allocation protect
00C7 148 DPT_STORE ORB,ORB$S_OWNER,AL,TTY$GL_OWNUIC ; Default owner UIC
00CE 149 DPT_STORE DDB,DDB$S_DDT,D,YF$DDT
00D3 150 DPT_STORE REINIT
00D3 151 DPT_STORE CRB,CRB$S_INTD+4,D,YF$INTINP ; RECEIVER INTERRUPT
00D8 152 DPT_STORE CRB,CRB$S_INTD2+4,D,YF$INTOUT ; TRANSMITTER INTERRUPT
00DD 153 DPT_STORE CRB,CRB$S_INTD+VEC$S_INITIAL,D,YF$INITIAL ; CONTROLLER INIT
00E2 154 DPT_STORE CRB,CRB$S_INTD+VEC$S_UNITINIT,D,YF$INITLINE; UNIT INIT
00E7 155
00E7 156 DPT_STORE END
0000 157
0000 158 DDTAB DEVNAM = YF,- ; DUMMY DHU PORT DRIVER DISPATCH TABLE
0000 159 START = 0,-
0000 160 FUNCTB = 0
00000038 161 .PSECT $$$115_DRIVER, LONG
0038 162
0038 163 ;
0038 164 ; THE ASSOCIATED CLASS DRIVER USES THIS TABLE TO COMMAND THE PORT DRIVER.
0038 165 ; THE ADDRESS OF THIS TABLE IS CONTAINED IN THE TERMINAL UCB EXTENSION AREA.
0038 166 ; THE OFFSET DEFINITIONS ARE DEFINED BY TTYDEFS.
0038 167 ;
00000000 0038 168 YF$SIL_ERROR::
0038 169 .LONG 0 ; Indicates Silo not empty at interrupt
00000000 003C 170 YF$S_ERROR::
0040 171 .LONG 0 ; Indicates DMA count non zero at
; interrupt
00000000 0040 173 YF$S_DMAXMT_ERROR::
0040 174 .LONG 0 ; Indicates DMA error bit set by DHU
0044 175 YF$S_INACT_ERROR::
```



```
00000000 0044 176 .LONG 0 ; Indicates UCB$M_INT clear at interrupt
          0048 177 PORT_VECTOR:
          0048 178
          0048 179 ;
          0048 180 ; DHU/DHV SPECIFIC DISPATCH TABLE
          0048 181 ;
          0048 182 $VECINI YF,YF$NULL
          0080 183 $VEC STARTIO,YF$STARTIO ; START NEW OUTPUT
          004C 184 $VEC SET_LINE,YF$SET_LINE ; SET NEW SPEED/PARITY
          0054 185 $VEC DS_SET,YF$DS_SET ; SET OUTPUT MODEM SIGNALS
          0058 186 $VEC XON,YF$XON ; SEND XON SEQUENCE
          005C 187 $VEC XOFF,YF$XOFF ; SEND XOFF SEQUENCE
          0060 188 $VEC STOP,YF$STOP ; STOP OUTPUT
          0064 189 $VEC ABORT,YF$ABORT ; ABORT OUTPUT IN PROGRESS
          006C 190 $VEC RESUME,YF$RESUME ; RESUME STOPPED OUTPUT
          0070 191 $VEC MAINT,YF$MAINT ; INVOKE MAINTENANCE FUNCTION
          007C 192 $VEC FORKRET,YF$FORK ; PORT FORK CALLBACK
          0080 193 $VECEND
          0084 194
          0084 195 ;
          0084 196 YF$NULL: ; NULL PORT ROUTINE
05 0084 197 RSB
    0085 198
```



```
0085 200 .SBTTL REGISTER DEFINITIONS
0085 201
0085 202
00000000 0085 203 DHUCSR = 0 ; Base CSR
00000002 0085 204 DHURBF = 2 ; Received data fifo
00000002 0085 205 DHUTXC = 2 ; DHV single char register
00000002 0085 206 DHUTCX = 2 ; DHU timer control
00000004 0085 207 DHULPR = 4 ; Line parameters
00000006 0085 208 DHUTXF = 6 ; Transmit FIFO
00000006 0085 209 DHUTFS = 6 ; Transmit FIFO size
00000007 0085 210 DHUSTT = 7 ; Modem status
00000008 0085 211 DHULCT = 8 ; Line control
0000000A 0085 212 DHUTBF1 = 10 ; low bits of address
0000000C 0085 213 DHUTBF2 = 12 ; High bits of address
0000000E 0085 214 DHUTCT = 14 ; Byte count
0085 215
0085 216
0085 217 : CSR BIT DEFINITIONS ( CSR ) ( READ/WRITE )
0085 218 : (NOTE: THIS REGISTER MUST ONLY BE READ IN RESPONSE TO A TRANSMIT INTERRUPT.
0085 219 : TO LOAD AN INDIRECT REGISTER, ONLY WRITE OPERATIONS MAY BE USED.
0085 220 : READ-MODIFY-WRITE OPERATIONS MUST NEVER BE USED)
0085 221
0085 222 $YIELD DHUCSR,0,<-
0085 223 <IADDR,4,M>,- ; INDIRECT REGISTER ADDRESS
0085 224 <1>,- ; DON'T USE THIS BIT
0085 225 <CLEAR,1,M>,- ; MASTER RESET
0085 226 <RCVINT,1,M>,- ; RECEIVER INTERRUPT ENABLE
0085 227 <1>,- ; DON'T USE THIS BIT
0085 228 <LINE,4,M>,- ; LINE NUMBER (0 - 15)
0085 229 <DMAERR,1,M>,- ; DMA error
0085 230 <DIGFAL,1,M>,- ; Diagnostic failed
0085 231 <SNDINT,1,M>,- ; TRANSMIT INTERRUPT ENABLE
0085 232 <SNDRDY,1,M>,- ; TRANSMITTER READY
0085 233 >
0085 234
0085 235 : RECEIVER BUFFER ( CSR+2 ) ( READ ONLY )
0085 236
0085 237 $YIELD DHURCV,0,<-
0085 238 <BUF,8,M>,- ; RECEIVER DATA
0085 239 <LINE,4,M>,- ; LINE NUMBER (0 - 7)
0085 240 <PARERR,1,M>,- ; PARITY ERROR
0085 241 <FRAMER,1,M>,- ; FRAME ERROR
0085 242 <OVERRUN,1,M>,- ; OVERRUN ERROR
0085 243 <VALID,1,M>,- ; DATA VALID
0085 244 >
0085 245
0085 246 : LINE PARAMETER REGISTER ( CSR+4 , Indirect register)
0085 247
0085 248
0085 249 $YIELD DHULPR,0,<-
0085 250 <1>,- ; DON'T USE THIS BIT
0085 251 <DIAG1,1,M>,- ; Diagnostics control
0085 252 <DIAG2,1,M>,- ; Diagnostics control
0085 253 <SIZE,2,M>,- ; CHARACTER SIZE
0085 254 <PARITY,1,M>,- ; PARITY ENABLE
0085 255 <ODD,1,M>,- ; ODD PARITY
0085 256 <STOP,1,M>,- ; NUMBER STOP BITS
```



```
0085 257 <RSPEED,4,M>,- : RECEIVER LINE SPEED
0085 258 <TSPEED,4,M>,- : TRANSMITTER LINE SPEED (BOTH RVC/TX FOR LINES 2-7)
0085 259 >
0085 260 :
0085 261 : LINE CONTROL INDIRECT REGISTER (CSR + 8, Indirect register)
0085 262 :
0085 263 :
0085 264 $VIELD DHULCT,0,<-
0085 265 <ABORT,1,M>,- : Output abort
0085 266 <OAUTO,1,M>,- : AUTO XON/OFF
0085 267 <RCV,1,M>,- : RECEIVER ENABLE
0085 268 <BREAK,1,M>,- : SEND BREAK
0085 269 <IAUTO,1,M>,- : Incoming Auto flow enable
0085 270 <SNDOFF,1,M>,- : Send XOFF
0085 271 <MAINT,2,M>,- : Maintenance
0085 272 <MODEM,1,M>,- : Modem control line
0085 273 <DTR,1,M>,- : Data terminal ready
0085 274 <2>,-
0085 275 <RTS,1,M>,- : Request to send
0085 276 >
0085 277 :
0085 278 : Status register
0085 279 $VIELD DHUSTT,0,<-
0085 280 <3>,-
0085 281 <CTS,1,M>,-
0085 282 <DCD,1,M>,-
0085 283 <RI,1,M>,-
0085 284 <1>,-
0085 285 <DSR,1,M>,-
0085 286 >
0085 287 :
0085 288 : Constant value for Base CSR
0085 289 : Enables both receive and transmit interrupts
0085 290
00004040 0085 291 DHUCSR$C_BASE = <DHUCSR$M_RCVINT!DHUCSR$M_SNDINT>
0085 292 :
0085 293 : MACRO USED TO ACCESS INDIRECT REGISTERS
0085 294 :
0085 295 :
0085 296 .MACRO SETIND REG
0085 297 .IF B, REG
0085 298 MOVL UCBSL CRB(R5),R0 : GET CRB ADDRESS
0085 299 MOVL @CRB$C_INTD+VEC$C_IDB(R0),R0 : GET CSR ADDRESS
0085 300 BISW3 #<DHUCSR$C_BASE>,-
0085 301 UCBSW_UNIT(R5),DHUCSR(R0); SELECT INDIRECT FIELD
0085 302 .IFF
0085 303 BISW3 #<DHUCSR$C_BASE>,-
0085 304 UCBSW_UNIT(R5),DHUCSR(REG); SELECT INDIRECT FIELD
0085 305 .ENDC
0085 306 .ENDM SETIND
0085 307 :
0085 308 :
0085 309 : Extra field in UCB
0085 310 : This field could be removed if 2 spare bits can be found in the UCB for
0085 311 : the use of the port driver. (This would save 4 bytes from the declared length
0085 312 : of the UCB)
0085 313
```



```
00000134 0085 314 UCBSB_DHUFLG = UCBSB_TT_LENGTH ; Allocate at end of previous ucb
0085 315
0085 316 $VIELD UCB,0,<-
0085 317 <MAP,1,M>,- ; Map registers available if 1
0085 318 <DHU,1,M>,- ; DHU if 1, DHV if 0
0085 319 <XOFF,1,M>,- ; XOFF if 1, XON if 0
0085 320 >
0085 321
0085 322 : OUTPUT DELAY MACRO
0085 323 : TO RELEASE THE UNIBUS
0085 324
0085 325 .MACRO DELAY
0085 326 .REPEAT 3
0085 327 NOP
0085 328 .ENDR
0085 329 .ENDM
0085 330
```

```
0085 332 .sbttl SPEED CONVERSION TABLES
0085 333
0085 334 :
0085 335 : macro to generate table of acceptable speeds for DHU/DHV
0085 336 :
0085 337 .MACRO SPD CONV BAUD
0085 338 .=$YF$VMS_speeds+$TT$C_'baud'
0085 339 .byte dhuspdc_'baud'
0085 340 .=$YF$DHU_speeds+$DHU$C_'baud'
0085 341 .byte tt$C_'baud'
0085 342 .ENDM
0085 343
0085 344 :
0085 345 : speed values recognized by the DHU/DHV
0085 346 :
00000000 0085 347 DHUSPD$C_BAUD_50=0
00000001 0085 348 DHUSPD$C_BAUD_75=1
00000002 0085 349 DHUSPD$C_BAUD_110=2
00000003 0085 350 DHUSPD$C_BAUD_134=3
00000004 0085 351 DHUSPD$C_BAUD_150=4
00000005 0085 352 DHUSPD$C_BAUD_300=5
00000006 0085 353 DHUSPD$C_BAUD_600=6
00000007 0085 354 DHUSPD$C_BAUD_1200=7
00000008 0085 355 DHUSPD$C_BAUD_1800=8
00000009 0085 356 DHUSPD$C_BAUD_2000=9
0000000A 0085 357 DHUSPD$C_BAUD_2400=10
0000000B 0085 358 DHUSPD$C_BAUD_4800=11
0000000C 0085 359 DHUSPD$C_BAUD_7200=12
0000000D 0085 360 DHUSPD$C_BAUD_9600=13
0000000E 0085 361 DHUSPD$C_BAUD_19200=14
0000000F 0085 362 DHUSPD$C_BAUD_38400=15
0085 363
0085 364 ; Allocate and initialize table of speed values
0085 365 YF$VMS_SPEEDS:
0085 366 .REPEAT 16
0085 367 .BYTE -1 ; Initial to illegal value
FF 0085 368 .ENDR
0095 369 YF$DHU_SPEEDS:
0095 370 .REPEAT 16
0095 371 .BYTE 0 ; Initial to zero
00 0095 372 .ENDR
00A5 373
00A5 374 ; Now build up the table of acceptable speed values
00A5 375
00A5 376 SPD CONV BAUD_75
0097 377 SPD CONV BAUD_110
0098 378 SPD CONV BAUD_134
0099 379 SPD CONV BAUD_150
009A 380 SPD CONV BAUD_300
009B 381 SPD CONV BAUD_600
009C 382 SPD CONV BAUD_1200
009D 383 SPD CONV BAUD_1800
009E 384 SPD CONV BAUD_2000
009F 385 SPD CONV BAUD_2400
00A0 386 SPD CONV BAUD_4800
00A1 387 SPD CONV BAUD_9600
00A3 388 SPD CONV BAUD_19200
```



```
00A4 390          .SBTTL  CONTROLLER INITIALIZATION
00A4 391
00A4 392
00A4 393
00A4 394 :++
00A4 395 : YF$DELIVER - Unit delivery routine
00A4 396 :
00A4 397 : FUNCTIONAL DESCRIPTION:
00A4 398 :
00A4 399 : THIS ROUTINE IS ENTERED AT SYSTEM STARTUP
00A4 400 :
00A4 401 : It checks the device to see if the specified unit number exists.
00A4 402 : This is necessary as the DHV has 8 lines, while the DHU has 16
00A4 403 :
00A4 404 : INPUTS:
00A4 405 :
00A4 406 :     R4 = ADDRESS OF THE CONTROLLER CSR
00A4 407 :     R5 = Unit number
00A4 408 :
00A4 409 : OUTPUTS:
00A4 410 :
00A4 411 :     R0 = 1 if unit exists
00A4 412 :         = 0 if unit does not exist
00A4 413 :
00A4 414 : IMPLICIT INPUTS:
00A4 415 :
00A4 416 :
00A4 417 :--
00A4 418 YF$DELIVER::
10  55  D1 00A4 419          CMPL      R5,#16
      1C  18 00A7 420          BGEQ     10$          ; exit if unit too large
      50  64 B0 00A9 421          MOVW     (R4),R0      ; Read base CSR
15  50  05  E0 00AC 422          BBS      #DHUCSR$V_CLEAR,R0,10$ ; controller bad
11  50  0D  E0 00B0 423          BBS      #DHUCSR$V_DIGFAL,R0,10$ ; controller bad
      00B4 424 :
      00B4 425 : Note. If controller has not passed self test then we cannot rely on
      00B4 426 : the value read from DHUSTT
      00B4 427 :
50  07  A4  90 00B4 428          MOVW     DHUSTT(R4),R0      ; get status byte
      05  50  E8 00B8 429          BLBS     R0,5$          ; low bit indicates DHU
      08  55  D1 00BB 430          CMPL     R5,#8          ; DHV has only 8 lines
      05  18 00BE 431          BGEQ     10$
      00C0 432 5$:
      50  01  D0 00C0 433          MOVL     #1,R0          ; unit exists
      02  11 00C3 434          BRB      20$
      00C5 435 10$:
      50  04 00C5 436          CLRL     R0
      00C7 437 20$:
      05  00C7 438          RSB
```

```
00C8 440 .SBTTL CONTROLLER INITIALIZATION
00C8 441
00C8 442
00C8 443
00C8 444 :++
00C8 445 : YF$INITIAL - INITIALIZE INTERFACE
00C8 446 :
00C8 447 : FUNCTIONAL DESCRIPTION:
00C8 448 :
00C8 449 : THIS ROUTINE IS ENTERED AT SYSTEM STARTUP AND POWER RECOVERY.
00C8 450 :
00C8 451 : INPUTS:
00C8 452 :
00C8 453 : R4 = ADDRESS OF THE UNIT CSR
00C8 454 : R5 = IDB OF UNIT
00C8 455 : R8 = ADDRESS OF THE UNIT CRB
00C8 456 :
00C8 457 : OUTPUTS:
00C8 458 :
00C8 459 : R2 is destroyed.
00C8 460 :
00C8 461 : IMPLICIT INPUTS:
00C8 462 :
00C8 463 : IPL = IPL$ POWER
00C8 464 :
00C8 465 :--
00C8 466 YF$INITIAL:: ; INITIALIZE DHU UNIT
00C8 467 :
00C8 468 : SET UP CONTROLLER
00C8 469 :
00C8 470 : CLASS_CTRL_INIT YF$DPT,PORT_VECTOR; RELOCATE THE NECESSARY TABLES
00F5 471 25$:
00F5 472 :
00F5 473 : Note. The DHV takes about 2 seconds to initialise
00F5 474 : and the DHU up to 5
00F5 475 : so we assume that initialization has taken place
00F5 476 : We could start the initialization and then do a 'skip self test'
00F5 477 : operation to bring the self test time down to a few milliseconds,
00F5 478 : however we would not then know if the board was good or not.
00F5 479 :
00F5 480 :
64 20 B3 00F5 481 BITW #DHUCSR$M_CLEAR,(R4)
00F5 482 BNEQ 26$ ; dont do reset if still there
64 20 B0 00FA 483 MOVW #DHUCSR$M_CLEAR,(R4) ; CONTROLLER RESET
00FD 484 26$:
00FD 485 :
00FD 486 : WAIT TILL CONTROLLER INITIALIZATION IS COMPLETE
00FD 487 :
00FD 488 : 5 second wait here !!!
00FD 489 :
00FD 490 TIMEWAIT #500000,#DHUCSR$M_CLEAR,(R4),W,.FALSE.
0124 491 :
0124 492 :
64 2E 50 E9 0124 493 BLBC R0,YF$CTRL_ERROR
64 2000 8F B3 0127 494 BITW #DHUCSR$M_DIGFAL,(R4)
00F5 495 BEQL 90$
50 00 B0 012E 496 MOVW #0,R0 ; failed self test, dont use
```



```

      22 11 0131 497 BRB YF$CTRL_ERROR
      64 4040 8F B0 0133 498 903:
      0133 499 MOVW #DHUCSR$C_BASE,(R4) ; set up base csr value
      0138 500
      0138 501
      0138 502 100$:
0B A8 46 8F 90 0138 503 MOVB #DT$_DHV,CRB$B_TT_TYPE(R8) ; CONTROLLER IS DHV
50 07 A4 90 013D 504 MOVB DHUSTT(R4),R0 ; test for DHU or DHV
0D 50 E9 0141 505 BLBC R0,110$ ; DHV does not have silo timeout
02 A4 00000000'GF 90 0144 506 MOVB G^TTY$GB_SILOTIME,DHUTCR(R4) ; INIT INPUT SILO TIMEOUT VALUE
0B A8 47 8F 90 014C 507 MOVB #DT$_DHU,CRB$B_TT_TYPE(R8) ; CONTROLLER IS DHU
50 01 D0 0151 508 110$:
05 05 0151 509 MOVL #SS$_NORMAL,R0
0154 510 RSB
0155 511
0155 512 YF$CTRL_ERROR:
05 0155 513 RSB
0156 514
```

```
0156 516 .SBTTL UNIT INITIALIZATION
0156 517
0156 518 : YF$INITLINE - UNIT INITIALIZATION
0156 519 :
0156 520 : FUNCTIONAL DESCRIPTION:
0156 521 :
0156 522 : THIS ROUTINE PERFORMS A SIMPLE UNIT INITIALIZATION.
0156 523 :
0156 524 : INPUTS:
0156 525 :
0156 526 : R5 = UCB ADDRESS
0156 527 :
0156 528 : OUTPUTS:
0156 529 :
0156 530 : R4,R5 ARE PRESERVED.
0156 531 :--
0156 532
0156 533 YF$INITLINE::
50 FEEF CF DE 0156 534 MOVAL YF$VEC,R0 ; GET THE DISPATCH TABLE ADDRESS
64 A5 10 A8 015B 535 CLASS_UNIT_INIT
53 01 54 A5 78 01A8 536 BISW #UCB$M_ONLINE,UCB$W_STS(R5); SET ONLINE
0106 C5 53 B0 01AD 537
OOF8 C5 00000000'GF 90 01B2 538 ASHL UCB$W_UNIT(R5),#1,R3 ; BUILD UNIT'S BIT MASK
0122 C5 A8 01BB 539 MOVW R3,UCB$W_TT_UNITBIT(R5) ; SAVE IT
51 0114 C5 D0 01BD 540 MOVW GATTY$GB_PARITY,UCB$B_TT_PARITY(R5) ; STORE TERMINAL'S PARITY VALUE IN UCB
08 B1 16 01C0 541 BISW #TTY$M_PC_DMAAVL!TTY$M_PC_XOFAVL,-; SHOW DMA FEATURE AVAILABLE FOR U
01C8 542 UCB$W_TT_PRTCTL(R5) ; IN PORT LEVEL
01C8 543
01C8 544
01C8 545 MOVL UCB$L_TT_CLASS(R5),R1 ; ADDRESS CLASS VECTOR TABLE
01C8 546 JSB @CLASS_SETUP_UCB(R1) ; INIT UCB FIELDS
01C8 547
01C8 548 :
01C8 549 : Perform check to see if device is good
01C8 550 :
54 24 30 BB 01C8 551 PUSHR #M<R4,R5>
54 2C B4 D0 01CA 552 MOVL UCB$L_CRB(R5),R4 ; GET CRB ADDRESS
55 54 A5 D0 01CE 553 MOVL @CRB$C_INTD+VEC$L_IDB(R4),R4 ; GET CSR ADDRESS
FECA CF 16 01D2 554 MOVZWL UCB$W_UNIT(R5),R5
04 30 BA 01D6 555 JSB YF$DECIVER ; test this device is ok
04 50 E8 01DA 556 POPR #M<R4,R5>
01DF 557 BLBS R0,11$
01DF 558 :
01DF 559 : DHU is in a bad way, set device offline
01DF 560 :
64 A5 10 AA 01DF 561 BICW #UCB$M_ONLINE,UCB$W_STS(R5); SET OFFLINE
01E3 562 11$:
01E3 563 : Test to see the type of configuration
01E3 564 :
01E7 565 CLRB UCB$B_DHUFLG(R5)
01F6 566 SETIND
0134 C5 94 01F6 567 BISB #UCB$M_DHU,UCB$B_DHUFLG(R5); Assume DHU
0134 C5 02 88 01FB 568 MOVW DHUSTT(R0),R0 ; test if DHU or DHV
50 07 A0 90 01FF 569 BLBS R0,12$
09 50 E8 0202 570 :
0202 571 : Note. The DHV always interrupts at BR4
0202 572 : This becomes IPL 20 on a VAX
```



```

0134 05 14 90 0202 573 ;
SE A5 02 8A 0202 574 ; MOVW #20,UCBSB_DIPL(R5)
0134 C5 02 8A 0206 575 BICB #UCBSM_DHO,UCBSB_DHUFLG(R5); Actually its a DHV
0208 576
0208 577 12$:
0208 578
0208 579 ; Find out what kind of machine the device is connected to.
0208 580 ; This tells us what kind of mapping registers we have
0208 581
0208 582 CPUDISP <-
0208 583 YF_INITMAP,- ; 11/780
0208 584 YF_INITMAP,- ; 11/750
0208 585 YF_INITMAP,- ; 11/730
0208 586 YF_INITMAP,- ; 11/790
0208 587 YF_INITNULL,- ; unknown
0208 588 YF_INITNULL,- ; unknown
0208 589 YF_NOMAP,- ; Seahorse
0208 590 YF_INITMAP- ; Mayflower
0208 591 >
0227 592 ; Unknown processor type
0227 593 YF_INITNULL:
0227 594
0227 595 ; prevent DMA being used, because we do not understand how the adapter
0227 596 ; works.
0227 597
0122 04 AA 0227 598 BICW #TTY$M_PC_DMAVL,-; SHOW DMA FEATURE NOT AVAILABLE FOR USE
C5 0229 599 UCBSW_TT_PRTCTL(R5) ; IN PORT LEVEL
022C 600
07 11 022C 601 BRB INIT_CONTINUE
022E 602 ; These processors have map registers for DMA operation
022E 603 YF_INITMAP:
022E 604
0134 C5 01 88 022E 605 BISB #UCBSM_MAP,UCBSB_DHUFLG(R5)
00 11 0233 606 BRB INIT_CONTINUE
0235 607
0235 608 ; These processors do not have map registers
0235 609
0235 610 YF_NOMAP:
0235 611 ; No bits set
0235 612
0235 613 INIT_CONTINUE:
0235 614
0235 615 ;SET MODE CODE NEEDS TO TOGGLE THESE BITS
0235 616
000009EB'EF 16 0235 617 JSB YF$SET_LINE ; INIT SPEED/PARITY
023B 618
023B 619
023B 620 ; ENABLE LINE RECEIVER , TRANSMITTER AND MODEM INTERRUPTS
023B 621
023B 622
23 64 A5 04 E1 023B 623 BBC #UCBSV_ONLINE,UCBSW_STS(R5),20$ ; TEST ONLINE
0240 624 SETIND R4
0D A4 94 0247 625 CLRB DHUTBF2+1(R4) ; assume transmitter should
024A 626 ; be disabled
08 A4 01 A8 024A 627 BISW #DHULCT$M_ABORT,DHULCT(R4) ; assume abort to be set
024E 628
OF 012A C5 07 E0 024E 629 BBS #UCBSV_TT_DSBL,UCBSB_TT_MAINT(R5),20$ ; Test if disabled
```

```
08 A4 0104 8F A8 0254 630
08 A4 01 AA 0254 631 BISW #DHULCTSM_RCV!DHULCTSM_MODEM,DHULCT(R4) ; enable receiver and modem
0D A4 80 8F 90 025A 632 BICW #DHULCTSM_ABORT,DHULCT(R4) ; clear abort bit
025E 633 MOVW #^X80,DHUTBF2+1(R4) ; enable transmitter
0263 634 20$:
0263 635
0263 636 : INIT RECEIVER MODEM STATUS FOR DHU
0263 637
0263 638
50 07 A4 90 0263 639 MOVW DHUSTT(R4),R0
0267 640
0267 641 : Unfortunately the DHU/DHV dont put the modem bits where we want them
0267 642 : To get the correct bits we have to move RI,DCD,CTS up one place, while not
0267 643 : changing DSR. spare bits have to be ignored.
50 50 50 C0 0267 644 ADDL R0,R0 ; (This shifts DSR out of bottom byte)
50 8F 8F 8A 026A 645 BICB #^C<TTSM_DS_RING!TTSM_DS_CTS!TTSM_DS_CARRIER>,R0
0124 C5 50 90 026E 646 MOVW R0,UCBSB-TT_DS_RCV(R5) ; UPDATE CURRENT INPUT MODEM SIGNALS
06 50 08 E1 0273 647 BBC #DHUSTT$V_DSR+T,R0,25$
00 0124 C5 07 E2 0277 648 BBSS #TT$V_DS_DSR,UCBSB-TT_DS_RCV(R5),25$
027D 649 25$:
50 51 00 9A 027D 650 MOVZBL #MODEMSC_INIT,R1 ; ASSUME INIT MODEM PROTOCOL
50 0114 C5 D0 0280 651 MOVL UCBSL TT_CLASS(R5),R0 ; ADDRESS CLASS VECTOR TABLE
0C B0 16 0285 652 JSB @CLASS_DS_TRAN(R0) ; INVOKE TO INIT MODEM PROTOCOL
0288 653 30$:
08 64 A5 05 E1 0288 654 BBC #UCBSV_POWER,UCBSW_STS(R5),40$ ; DID WE DETECT A POWER FAIL
50 0114 C5 D0 028D 655 MOVL UCBSL TT_CLASS(R5),R0 ; GET THE CLASS VECTOR TABLE ADDRESS
20 B0 17 0292 656 JMP @CLASS_POWERFAIL(R0) ; AND GOTO THE POWERFAIL CODE
0295 657
05 0295 658 40$: RSB
0296 659
0296 660 : ERROR DETECTED DURING INITIALIZATION
0296 661
0296 662
64 A5 10 AA 0296 663 YF$UNIT_ERROR:
05 029A 664 BICW #UCBSM_ONLINE,UCBSW_STS(R5) ; UNIT NOT ON LINE
029B 665 RSB
029B 666
```



```
029B 668 .SBTTL MAINTENANCE ROUTINES
029B 669 :++
029B 670 : YFSMAINT - MAINTENANCE FUNCTIONS
029B 671 :
029B 672 : FUNCTIONAL DESCRIPTION:
029B 673 : THIS ROUTINE PERFORMS MAINTENANCE FUNCTIONS FOR THE DHU
029B 674 : (LOOPBACK IS ONLY ALLOWED ON LINES 0 AND 1)
029B 675 :
029B 676 : INPUTS:
029B 677 :
029B 678 : R5 = UCB ADDRESS
029B 679 : UCB$B_TT_MAINT = FUNCTION TO BE PERFORMED
029B 680 :
029B 681 : OUTPUTS:
029B 682 : R0-R4 SCRATCH
029B 683 : --
029B 684 :
029B 685 YFSMAINT:
029B 686 BITB #IOSM_LOOPa-7,- ; LOOPBACK FUNCTION
029B 687 UCB$B_TT_MAINT(R5)
029B 688 BEQL 5$ ; NO
029B 689 MOVZWL #^X02,R2 ; SPECIFY LOOPBACK CODE
029B 690 BRB 10$
029B 691 5$:
029B 692 BITB #IOSM_UNLOOPa-7,- ; RESET LOOPBACK FUNCTION
029B 693 UCB$B_TT_MAINT(R5)
029B 694 BEQL 15$ ; NO
029B 695 MOVZWL #^X00,R2 ; SPECIFY UNLOOP CODE
029B 696 10$:
029B 697 SETIND
029B 698 MOVW DHULCT(R0),R1
029B 699 INSV R2,#DHULCT$V_MAINT,#2,R1 ; SET MAINT FIELD
029B 700 MOVW R1,DHULCT(R0) ; Update
029B 701 MOVZBL #1,R0 ; INDICATE SUCCESS
029B 702 RSB
029B 703 50$:
029B 704 CLRL R0
029B 705 RSB
029B 706
029B 707 15$:
029B 708 BITB #IOSM_AUTXOF_ENAa-7,- ; AUTOXON ENABLED
029B 709 UCB$B_TT_MAINT(R5) ; NO THEN MAYBE DISABLE
029B 710 BEQL 17$
029B 711 BISW #TTY$M_PC_XOFAVL,- ; SET THE BIT AVAILABLE
029B 712 UCB$W_TT_PRTCTL(R5)
029B 713
029B 714 17$:
029B 715 BITB #IOSM_AUTXOF_DISa-7,- ; AUTOXON disabled
029B 716 UCB$B_TT_MAINT(R5) ; no then don't disable it
029B 717 BEQL 19$
029B 718 BICW #TTY$M_PC_XOFAVL,-
029B 719 UCB$W_TT_PRTCTL(R5)
029B 720 19$:
029B 721 BITB #IOSM_LINE_OFFa-7,- ; LINE OFF
029B 722 UCB$B_TT_MAINT(R5) ; NO
029B 723 BEQL 30$
029B 724 SETIND
```

01 93 029B 686
012A C5 05 13 029D 687
52 02 3C 02A0 688
0A 11 02A2 689
02 93 02A5 690
012A C5 02 93 02A7 691
C5 02 93 02A7 692
52 26 13 02A9 693
00 3C 02AC 694
02AE 695
51 08 A0 B0 02B1 696
02 06 52 F0 02B1 697
08 A0 51 B0 02C0 698
50 01 9A 02C4 699
05 02D0 700
D4 02D1 701
05 02D1 702
02D3 703
02D4 704
40 8F 93 02D4 705
012A C5 05 13 02D4 706
05 13 02D7 707
20 A8 02DA 708
0122 C5 02DC 709
02DE 710
80 8F 93 02E1 711
012A C5 05 13 02E1 712
05 13 02E4 713
20 AA 02E7 714
0122 C5 02E9 715
02EB 716
02EE 717
02EE 718
012A 04 93 02EE 719
C5 05 13 02F0 720
1E 13 02F3 721
02F5 722

08 A0	0104 8F	AA 0304	725	BICW	#DHULCTSM_RCV!DHULCTSM_MODEM,DHULCT(R0) ; Disable Receive and modem
08 A0	01	A8 030A	726	BISW	#DHULCTSM_ABORT,DHULCT(R0) ; Abort any current activity
	0D A0	94 030E	727	CLRB	DHUTBF2+1(R0) ; Disable Transmit
	25	11 0311	728	BRB	40\$
			729		
	10	93 0313	730	BITB	#IOSM_LINE_ON@-7- ; LINE ON
012A	C5		731		UCB\$B_TT_MAINT(R5)
	B7	13 0315	732	BEQL	50\$; NO
			733	SETIND	
08 A0	0104 8F	A8 0329	734	BISW	#DHULCTSM_RCV!DHULCTSM_MODEM,DHULCT(R0) ; Enable Receive and modem
08 A0	01	AA 032F	735	BICW	#DHULCTSM_ABORT,DHULCT(R0)
0D A0	80 8F	90 0333	736	MOVB	#^X80,DHUTBF2+1(R0) ; Enable Transmit
	50 01	9A 0338	737	MOVZBL	#1,R0
		05 033B	738	RSB	
			739		
		033C	740		
		033C			


```
033C 742 .SBTTL OUTPUT MODEM CONTROL
033C 743 :++
033C 744 : YF$DS_SET - SET OUTPUT MODEM SIGNALS
033C 745 :
033C 746 : FUNCTIONAL DESCRIPTION:
033C 747 :
033C 748 : THIS ROUTINE OUTPUTS THE OUTPUT MODEM SIGNALS FOR THE SPECIFIED UNIT
033C 749 :
033C 750 : INPUTS:
033C 751 :
033C 752 :     R2 = LOW BYTE - SIGNALS TO ACTIVATE
033C 753 :     HIGH BYTE- SIGNALS TO DEACTIVATE
033C 754 :
033C 755 :     R5 = UCB ADDRESS
033C 756 :
033C 757 : OUTPUTS:
033C 758 :
033C 759 :     R0-R3 ARE USED.
033C 760 : --
033C 761 : YF$DS_SET:
033C 762 :
033C 763 : Check that the DHU/DHV modem control signals have the standard values
033C 764 :
033C 765 :     BISB R2,UCB$B_TT_DS_TX(R5) ; SET NEW OUTPUT SIGNALS
033C 766 :     ASHL #-8,R2,R2 ; ACCESS SIGNALS TO RESET
033C 767 :     BICB R2,UCB$B_TT_DS_TX(R5) ; RESET THEM
033C 768 :     SETIND
033C 769 :     BICB3 #^C<<DHULCT$M_DTR!DHULCT$M_RTS>/256>,UCB$B_TT_DS_TX(R5),-(SP);
0361 770 :
0361 771 : The DHU/DHV is set to report modem change events in the receive fifo
0361 772 : (Unless the line is disabled)
0361 773 :
0361 774 :     BBS #UCB$V_TT_DSBL,UCB$B_TT_MAINT(R5),10$ ; Test if disabled
0367 775 :     BISB #<DHULCT$M_MODEM/256>,(SP)
036A 776 10$:
036A 777 :     MOVB (SP)+,DHULCT+1(R0) ; OUTPUT NEW VALUE
036E 778 :
036E 779 :     RSB
036F 780 :
036F 781 :
```

0125 C5 52 88
52 52 F8 8F 78
0125 C5 52 8A
7E 0125 C5 ED 8F 8B
03 012A C5 07 E0
6E 01 88
09 A0 8E 90
05


```
036F 783 .SBTTL RECEIVER INTERRUPT SERVICE
036F 784 :++
036F 785 : YF$INTINP - DHU RECEIVER READY INTERRUPTS
036F 786 :
036F 787 : FUNCTIONAL DESCRIPTION:
036F 788 :
036F 789 : THIS ROUTINE IS ENTERED WHEN A CHARACTER IS AVAILABLE IN THE UNIT'S
036F 790 : SILO. THE CHARACTER IS EXTRACTED AND IS PASSED TO THE ASSOCIATED
036F 791 : CLASS DRIVER. IF THE CLASS DRIVER RETURNS CHARACTER(S) THEN NEW
036F 792 : OUTPUT IT INITIATED (NORMALLY ECHO).
036F 793 :
036F 794 : INPUTS:
036F 795 :
036F 796 :         00(SP) = ADDRESS OF IDB
036F 797 :
036F 798 : IMPLICIT INPUTS:
036F 799 :
036F 800 :         R0,R1,R2,R3,R4,R5 ARE SAVED ON STACK.
036F 801 :
036F 802 : OUTPUTS:
036F 803 :
036F 804 :         THE INTERRUPT IS DISMISSED WHEN THE SILO IS EMPTY.
036F 805 :
036F 806 :--
036F 807 YF$INTINP::                                ; DHU/DHV INPUT INTERRUPTS
036F 808 :
036F 809 : GET THE CSR ADDRESS
036F 810 :
036F 811 :         MOVL    @ (SP)+,R4                        ; GET THE IDB ADDRESS
036F 812 :         PUSHL   R4                                ; SAVE IDB ADDRESS
036F 813 :         MOVL    (R4),R0                            ; GET THE CSR ADDRESS
036F 814 :
036F 815 : GET THE CHARACTER FROM THE INTERFACE
036F 816 :
036F 817 25$:    MOVW    DHURBF(R0),R3                    ; Get the silo entry
036F 818 :         BGEQ     45$                                ; Silo empty (== BRW 100$)
036F 819 :         ASHL     #-8,R3,R2                          ; shift the line number
036F 820 :         BICL     #^C<15>,R2                        ; use mask to obtain line number
036F 821 :         MOVL     IDB$L_UCBLST(R4)[R2],R5            ; GET THE UCB FOR THAT LINE
036F 822 :         BEQL     25$                                ; IF EQL THEN NOT THERE
036F 823 :         BITW     #<DHURCVSM_PARERR>!--
036F 824 :         <DHURCVSM_OVERRUN>!--
036F 825 :         <DHURCVSM_FRAMER>,R3                        ; ERRORS OR MODEM TRANSITION ?
036F 826 :
036F 827 :         BNEQ     50$                                ; YES, PROCESS THEM
036F 828 27$:
036F 829 :         MOVZBL   R3,R3                                ; CLEAR THE HIGH BYTES OF CHARACTER
036F 830 :         JSB      @UCB$L_TT_PUTNXT(R5)                ; BUFFER THE CHARACTER
036F 831 :         BLEQ     40$                                ; NONE OR STRING OUTPUT
036F 832 :         TIMSET   #1,R1,LOCKOUTPUT                    ; SET TIMEOUT AND INTERRUPT BIT
036F 833 :         SETIND   R0
036F 834 :         BBS      #UCB$V_DHU,UCB$B_DHUFLG(R5),28$
036F 835 :         BISW3    #^X8000,R3,DHUTXC(R0)                ; DHV single char output
036F 836 :         BRB      30$
036F 837 28$:
036F 838 :         MOVW     R3,DHUTXF(R0)                        ; DHU fifo output
036F 839 30$:    MOVL     (SP),R4                        ; GET IDB ADDRESS
```



```

      97  11 03DE 840      BRB 25$      ; CONTINUE
      F9  13 03E0 841 40$:      BEQL 30$      ; NO CHARACTER
00000443'EF 16 03E0 842      JSB BURST_OUTPUT ; START BURST
      F1  11 03E2 843      BRB 30$
      03EA 844
      03EA 845
      18  11 03EA 846 45$:      BRB 100$
      03EC 847
      03EC 848
      03EC 849 50$:
      03EC 850
      03EC 851 : PROCESS PARITY, FRAME OVERRUN ERROR OR MODEM TRANSITION
      03EC 852
      03EC 853
      03EC 854 : The DHU indicates modem transition by setting all the
      03EC 855 : error bits
      08 53 0C E1 03EC 856      BBC #DHURCV$V_PARERR,R3,60$
      04 53 0E E1 03F0 857      BBC #DHURCV$V_OVERRUN,R3,60$
      19 53 0D E0 03F4 858      BBS #DHURCV$V_FRAMER,R3,200$; Modem transition if all set
      03F8 859 60$:
      52 0114 C5 D0 03F8 860      MOVL UCB$L TT_CLASS(R5),R2 ; GET CLASS DISPATCH
      14 B2 16 03FD 861      JSB @CLASS_READERROR(R2) ; SIGNAL ERROR
      95 12 0400 862      BNEQ 27$ ; CHRRACTER TO ECHO
      0402 863 70$:
      D7 11 0402 864      BRB 30$
      0404 865
      5E 04 C0 0404 866 100$: ADDL #4,SP ; REMOVE IDB ADDRESS
      50 8E 7D 0407 867      MOVQ (SP)+,R0 ; RESTORE REGISTERS
      52 8E 7D 040A 868      MOVQ (SP)+,R2
      54 8E 7D 040D 869      MOVQ (SP)+,R4
      02 0410 870      REI
      0411 871
      0411 872 : Modem transition routine
      0411 873
      0411 874 200$:
      0411 875 : If lsb set then it is a self test code
      0411 876 : We currently ignore self test codes, although these could be used
      0411 877 : for error logging purposes
      0411 878
      C7 53 E8 0411 879      BLBS R3,30$
      0414 880
      0414 881 : Unfortunately the DHU and DHV dont put the modem bits where we want them
      0414 882 : To get the correct bits we have to move RI,DCD,CTS up one place, while not
      0414 883 : changing DSR. spare bits have to be ignored.
      53 53 53 C0 0414 884      ADDL R3,R3 ; (This shifts DSR out of bottom byte)
      53 8F 8F 8A 0417 885      BICB #^C<TT$M_DS_RING!TT$M_DS_CTS!TT$M_DS_CARRIER>,R3
      04 53 08 E1 041B 886      BBC #DHUSTT$V_DSR+1,R3,210$
      00 53 07 E2 041F 887      BBSS #TT$V_DS_DSR,R3,210$
      0423 888 210$:
      0124 C5 53 90 0423 889      MOVB R3,UCB$B_TT_DS_RCV(R5) ; UPDATE CURRENT INPUT MODEM SIGNALS
      52 53 90 0428 890      MOVB R3,R2 ; PASS CURRENT INPUT MODEM SIGNALS IN R2
      51 03 9A 042B 891      MOVZBL #MODEM$C_DATASET,R1 ; TRANSITION TYPE IS DATASET
      50 DD 042E 892      PUSHL R0 ; SAVE CSR ADDRESS
      54 0114 C5 D0 0430 893      MOVL UCB$L TT_CLASS(R5),R4 ; GET CLASS DISPATCH
      0C B4 16 0435 894      JSB @CLASS_DS_TRAN(R4) ; INVOKE TRANSITION ROUTINE
      50 8ED0 0438 895      POPL R0 ; RESTORE CSR ADDRESS
      FF9D 31 043B 896      BRW 30$
```

YFDRIVER
V04-000

- Port Driver for DHU/DHV
RECEIVER INTERRUPT SERVICE

F 7

16-SEP-1984 02:26:48
5-SEP-1984 04:17:43

VAX/VMS Macro V04-00
[TTDRVR.SRC]YFDRIVER.MAR;1

Page 21
(1)

043E 897
043E 898

YFDI
V04-


```
043E 900 .SBTTL START I/O ROUTINE
043E 901 :++
043E 902 :YF$STARTIO - START I/O OPERATION ON DHU
043E 903 :
043E 904 :FUNCTIONAL DESCRIPTION:
043E 905 :
043E 906 :THIS ROUTINE IS ENTERED FROM THE DEVICE INDEPENDENT TERMINAL STARTIO
043E 907 :ROUTINE TO ENABLE OUTPUT INTERRUPTS ON AN IDLE DHU UNIT.
043E 908 :
043E 909 :INPUTS:
043E 910 :
043E 911 :R3 = CHARACTER AND CC = PLUS
043E 912 :ADDRESS AND CC = NEGATIVE
043E 913 :
043E 914 :R5 = UCB ADDRESS
043E 915 :
043E 916 :OUTPUTS:
043E 917 :
043E 918 :R5 = UCB ADDRESS
043E 919 :--
043E 920 :.ENABLE LSB
043E 921 :YF$STARTIO:: ; START I/O ON UNIT
043E 922 :BLSS BURST_OUTPUT
043E 923 :BRW 90$
043E 924 :
043E 925 :BURST_OUTPUT:
043E 926 :SETIND
043E 927 :MOVZWL UCBSW TT_OUTLEN(R5),R2 ; GET LENGTH
043E 928 :BBC #TTY$V PC_DMAENA,- ; USE SILO IF DMA NOT ENABLED ON THIS LINE
043E 929 :UCBSW TT_PRTCTL(R5),SILO_OUTPUT
043E 930 :CMPW R2,G*TTY$GW-DMA SIZE ; LARGE ENOUGH FOR DMA
043E 931 :BLSS SILO_OUTPUT
043E 932 :BRW DMA_START ; YES SO DO DMA
043E 933 :
043E 934 :SILO_OUTPUT:
043E 935 :BBC #UCBSV DHU,UCBSB_DHUFLG(R5),200$ ; DHV has single char output
043E 936 :MOVZBL DHUTFSR0,R1 ; get number of slots
043E 937 :CMPW R2,R1 ; BURST LARGER THAN SILO?
043E 938 :BLEQU 50$ ; NO
043E 939 :MOVZBL R1,R2 ; SLOTS AVAILABLE IS MAXIMUM
043E 940 :50$:
043E 941 :MOVL UCBSL TT_OUTADR(R5),R3 ; GET ADDRESS
043E 942 :ADDL R2,UCBSL TT_OUTADR(R5) ; UPDATE POINTER
043E 943 :SUBW R2,UCBSW TT_OUTLEN(R5) ; AND COUNT
043E 944 :BEQL 60$ ; ALL DONE, NO NEED FOR BURST
043E 945 :BISW #TTY$M TANK BURST,- ; SIGNAL BURST ACTIVE
043E 946 :UCBSW TT_HOCD(R5)
043E 947 :60$:
043E 948 :BLBC R2,70$ ; EVEN TRANSFER
043E 949 :MOVB (R3)+,DHUTXF(R0) ; OUTPUT ODD BYTE
043E 950 :DECL R2 ; UPDATE COUNT
043E 951 :BEQL 80$ ; DONE
043E 952 :70$:
043E 953 :ASHL #-1,R2,R2 ; CONVERT TO WORD COUNT
043E 954 :75$:
043E 955 :MOVW (R3)+,DHUTXF(R0)
043E 956 :DELAY ; TO RELEASE THE UNIBUS
```

```
F6 52 F5 04AA 957 SOBGTR R2,75$ ; LOOP TILL DONE
      04AD 958
      05 04AD 959 80$: RSB ; RETURN TO CALLER
      04AE 960
      04AE 961 90$:
      22 13 04AE 962 BEQL 100$ ; SKIP IF NONE
      04B0 963 SETIND
09 0134 C5 01 E0 04BF 964 BBS #UCB$V_DHU,UCB$B_DHUFLG(R5),95$
      04C5 965 ;
      04C5 966 ; DHV single character output
      04C5 967 ;
02 A0 53 8000 8F A9 04C5 968 BISW3 #^X8000,R3,DHUTXC(R0)
      04 11 04CC 969 BRB 100$
      04CE 970 ;
      04CE 971 ; DHU fifo output
      04CE 972 ;
      04CE 973 95$:
      06 A0 53 90 04CE 974 MOVB R3,DHUTXF(R0)
      04D2 975 100$:
      05 04D2 976 RSB
      04D3 977 ;
      04D3 978 ; DHV 'silo' output, uses single character mode
      04D3 979 ;
      04D3 980 200$:
      53 011C D5 9A 04D3 981 MOVZBL @UCB$L_TT_OUTADR(R5),R3 ; GET character
      011C C5 D6 04D8 982 INCL UCB$L_TT_OUTADR(R5) ; UPDATE POINTER
      0120 C5 B7 04DC 983 DECW UCB$W_TT_OUTLEN(R5) ; AND COUNT
      07 13 04E0 984 BEQL 260$ ; ALL DONE, NO NEED FOR BURST
      0800 8F A8 04E2 985 BISW #TTY$M_TANK_BURST,- ; SIGNAL BURST ACTIVE
      0108 C5 04E6 986 UCB$W_TT_HOLD(R5)
      04E9 987 260$:
02 A0 53 8000 8F A9 04E9 988 BISW3 #^X8000,R3,DHUTXC(R0) ; output the character
      05 04F0 989 RSB
      04F1 990
      04F1 991 .DISABLE LSB
      04F1 992
```



```
04F1 994 .SBTTL PORT DMA ROUTINES
04F1 995 :++
04F1 996 DMA_START - INITIATE DMA OUTPUT
04F1 997
04F1 998 FUNCTIONAL DESCRIPTION:
04F1 999
04F1 1000 THESE ROUTINES ARE CALLED BY THE PORT INPUT INTERRUPT, OUTPUT
04F1 1001 INTERRUPT, AND STARTIO TO INITIATE NEW DMA OUTPUT.
04F1 1002 THEY HANDLE ALLOCATION AND LOADING OF MAP REGISTERS
04F1 1003 TO HANDLE DMA OUTPUT. MAP REGISTERS ARE ALLOCATED IN PAIRS
04F1 1004 TO ALLOW OUTPUT BURSTS UP TO 512 BYTES. TRANSFERS LARGER THAN
04F1 1005 THAT ARE DONE IN SEGMENTS. IF INSUFFICIENT MAPS ARE AVAILABLE,
04F1 1006 THE TRANSFER IS DONE SILO MODE.
04F1 1007
04F1 1008
04F1 1009 INPUTS:
04F1 1010
04F1 1011 R5 = UCB ADDRESS
04F1 1012
04F1 1013 OUTPUTS:
04F1 1014
04F1 1015 R5 = UCB ADDRESS
04F1 1016
04F1 1017 R0 PRESERVED
04F1 1018 R1,R2,R3,R4 DESTROYED
04F1 1019 --
04F1 1020 DMA_START:
04F1 1021 BISW #TTY$M_TANK_DMA, - ; SHOW DMA MODE ACTIVE
04F5 1022 UCBSW_TT_HOLD(R5)
04F8 1023 BBCC #TTY$V_TP_ABORT, - ; RESET ANY OLD ABORT REQUESTS
04FA 1024 UCBSB_TP_STAT(R5),3$
04FE 1025 BICW #DHULCT$M_ABORT, DHULCT(R0)
0502 1026
0502 1027 : If there are no mapping registers for the device then don't allocate them
0502 1028
0502 1029 3$: BBC #UCBSV_MAP, UCBSB_DHUFLG(R5),DMA_CONTINUE
0508 1030 : CHECK IF UNIT HAS PERMANENT MAP REGISTERS
0508 1031
0508 1032 BBC #TTY$V_PC_PRRMAP, - ; SKIP IF NOT AUTHORIZED FOR PERM MAPS
050A 1033 UCBSW_TT_PRTCTL(R5),5$
050E 1034 BBS #TTY$V_PC_MAPAVL, - ; SKIP FORK IF MAPS ALLOCATED ALREADY
0510 1035 UCBSW_TT_PRTCTL(R5),DMA_CONTINUE
0514 1036 5$:
0514 1037 BISB #TTY$M_TP_ALLOC, - ; SHOW ALLOC FORK ACTIVE
0516 1038 UCBSB_TP_STAT(R5)
0519 1039 MOVL UCBSL_TP_CLASS(R5),R1 ; GET CLASS VECTOR ADDRESS
051E 1040 JSB @CLASS_FORK(R1) ; FORK TO FIPL FOR MAP
0521 1041 ; REGISTER ALLOCATION
0521 1042 RSB ; RETURN TO CALLER WITH
0522 1043 ; "INT" LEFT ON TO INTERLOCK
0522 1044 ; OUTPUT. FORK ROUTINE WILL
0522 1045 ; RESUME AT DMA_ALLOC.
0522 1046 DMA_ALLOC:
0522 1047
0522 1048
0522 1049 5$:
0522 1050 MOVZBL #2,R3 ; REQUEST 2 MAP REGISTERS
```

1000 8F A8
0108 C5
00 E5
04 0130 C5
08 A0 01 AA

55 0134 C5 00 E1
06 0122 C5 03 E1
04 0122 C5 04 E0
02 0130 C5 88
51 0114 C5 D0
1C B1 16
05

53 02 9A

```
00000000*GF 16 0525 1051 JSB G*IOCSALOUBAMAPN ;
052B 1052
052B 1053 SETIPL UCBSB_DIPL(R5) ; INTERLOCK TO DEVICE IPL
0122 10 A8 052F 1054 BISW #TTY$M_PC_MAPAVL,- ; SHOW MAP ALLOCATED
02 0531 1055 UCBSW_TT_PRTCTL(R5)
0130 02 8A 0534 1056 BICB #TTY$M_TP_ALLOC,- ; SHOW ALLOC FORK DONE
05 0536 1057 UCBSB_TP_STAT(R5)
17 50 E8 0539 1058 BLBS R0,20$ ; SUCCESS
053C 1059
50 24 A5 D0 053C 1060 MOVL UCBSL_CRB(R5),R0 ; GET CRB OF UNIT
50 2C B0 D0 0540 1061 MOVL @CRBSL_INTD+VEC$SL_IDB(R0),R0 ; GET CSR
1000 8F AA 0544 1062 BICW #TTY$M_TANK_DMA,- ; RESET DMA MODE
0108 C5 0548 1063 UCBSW_TT_HOLD(R5)
52 0120 C5 3C 054B 1064 MOVZWL UCBSW_TT_OUTLEN(R5),R2 ; RESTORE OUTPUT LENGTH
FF16 31 0550 1065 BRW SILO_OUTPUT ; USE SILO FOR OUTPUT
0553 1066
50 24 A5 D0 0553 1067 20$: MOVL UCBSL_CRB(R5),R0 ; GET CRB ADDRESS
34 A0 D0 0557 1068 MOVL CRBSL_INTD+VEC$M_MAPREG(R0),- ;
012C C5 055A 1069 UCBSL_TP_MAP(R5) ; SAVE MAP FIELD IN UCB
055D 1070
055D 1071
055D 1072
055D 1073 DMA_CONTINUE:
0C 0130 C5 E0 055D 1074 BBS #TTY$V_TP_ABORT,- ; BRANCH IF DMA TO BE ABORTED
055F 1075 UCBSB_TP_STAT(R5),2$
0563 1076
53 011C C5 D0 0563 1077 MOVL UCBSL_TT_OUTADR(R5),R3 ; GET ADDRESS OF NEXT STRING
52 0120 C5 3C 0568 1078 MOVZWL UCBSW_TT_OUTLEN(R5),R2 ; LENGTH OF OUTPUT
03 12 056D 1079 BNEQ 4$ ; SKIP IF MORE TO DO
0162 31 056F 1080 2$: BRW DMA_DONE ; BRANCH IF TRANSFER IS DONE
50 DD 0572 1081 4$: PUSHL R0 ; SAVE INPUT VOLITAL REGISTER "CSR"
0574 1082
0574 1083
0574 1084 ; If there are no mapping registers for the device then
0574 1085 ; start dma directly
0574 1086
0574 1087
03 0134 C5 00 E1 0574 1088 BBC #UCBS$V_MAP, UCBSB_DHUFLG(R5),DMA_NOMAP
00A0 31 057A 1089 BRW DMA_MAP
```



```
057D 1091 DMA_NOMAP:
057D 1092 :
057D 1093 : Special code to work with an unmapped DHU/DHV
057D 1094 : (I.e. Seahorse configuration)
057D 1095 :
057D 1096 : this code examines the DMA buffer to find the maximum physically
057D 1097 : contiguous area, and starts a DMA on that part.
057D 1098 :
057D 1099 : It assumes that the buffer is in non-paged pool
057D 1100 : (I.e. that it is in system address space and that the PTE
057D 1101 : will always contain the PFN for the page)
057D 1102 :
057D 1103 : Find physical address of first page
057D 1104 :
54 53 53 011C C5 D0 057D 1105 :      MOVL      UCBSL TT_OUTADR(R5),R3
53 80000000 8F CB 0582 1106 :      BICL3     #^X80000000,R3,R4      : CALC SVAPTE OF BUFFER
54 54 F7 8F 78 058A 1107 :      ASHL      #-9,R4,R4              : ISOLATE PAGE
50 00000000 GF D0 058F 1108 :      MOVL      G^MMG$GL_SPTBASE,R0    : GETS SVAPTE OF BUFFER
54 54 6044 DE 0596 1109 :      MOVAL     (R0)[R4],R4            : INTO R4
53 FFFFE00 8F CA 059A 1110 :      BICL      #^C^X1FF,R3           : COMPUTE BYTE OFFSET IN PAGE
05A1 1111 :
05A1 1112 : calculate length of DMA in this page
05A1 1113 :
51 00000200 8F 53 C3 05A1 1114 :      SUBL3     R3,#512,R1
05A9 1115 :
05A9 1116 : See if this is longer than buffer
05A9 1117 :
0120 C5 51 B1 05A9 1118 :      CMPW      R1,UCBSW_TT_OUTLEN(R5)
05 05 15 05AE 1119 :      BLEQ      10$
51 0120 C5 B0 05B0 1120 :      MOVW      UCBSW_TT_OUTLEN(R5),R1 ; use actual buffer length
0120 C5 51 A2 05B5 1121 10$:
05B5 1122 :      SUBW      R1,UCBSW_TT_OUTLEN(R5)
05BA 1123 :
05BA 1124 : Get the PFN for the first page
05BA 1125 :
52 84 D0 05BA 1126 :      MOVL      (R4)+,R2
05BD 1127 :
05BD 1128 : calculate physical address from PTE
05BD 1129 : (assuming that it fits into a 22-bit address)
05BD 1130 :
52 52 15 00 EF 05BD 1131 :      EXTZV     #0,#21,R2,R2          : get PFN only
53 0D 09 52 F0 05C2 1132 :      INSV      R2,#9,#13,R3
05C7 1133 :
05C7 1134 : Loop through remaining pages to see if they are contiguous with this one
05C7 1135 :
05C7 1136 20$:
0120 C5 B5 05C7 1137 :      TSTW      UCBSW_TT_OUTLEN(R5)
05 2D 13 05CB 1138 :      BEQL      50$                  : No more DMA
05 52 D6 05CD 1139 :      INCL      R2                    : get expected PTE
52 64 15 00 ED 05CF 1140 :      CMPZV     #0,#21,(R4),R2       : test next PTE
05 24 12 05D4 1141 :      BNEQ      50$                  : not contiguous
05 84 D5 05D6 1142 :      TSTL      (R4)+                 : step to next PTE
05D8 1143 :
05D8 1144 : concatenate this page to previous DMA
05D8 1145 :
0200 8F 0120 C5 B1 05D8 1146 :      CMPW      UCBSW_TT_OUTLEN(R5),#512
05 10 15 05DF 1147 :      BLEQ      40$                  : partial page
```

```

51 00000200 8F C0 05E1 1148
0120 C5 0200 8F A2 05E1 1149 ADDL #512,R1 ; whole page gets added
D6 11 05E8 1150 SUBW #512,UCBSW_TT_OUTLEN(R5)
05EF 1151 BRB 20$
05F1 1152
05F1 1153 ; partial page gets added, to complete the DMA
05F1 1154 40$:
51 0120 C5 A0 05F1 1155 ADDW UCBSW_TT_OUTLEN(R5),R1
0120 C5 B4 05F6 1156 CLRW UCBSW_TT_OUTLEN(R5)
05FA 1157 50$:
05FA 1158
05FA 1159 : Start up the DMA
05FA 1160
011C C5 51 C0 05FA 1161 ADDL R1,UCBSL_TT_OUTADR(R5) ; step to next DMA address
50 8ED0 05FF 1162 POPL R0 ; RESTORE CSR ADDRESS
0602 1163 SETIND R0
OE A0 51 B0 0609 1164 MOVW R1,DHUTCT(R0) ; load the count
OA A0 53 B0 060D 1165 MOVW R3,DHUTBF1(R0) ; load low address
53 53 06 10 EF 0611 1166 EXTZV #16,#6,R3,R3 ; get high address
OC A0 53 80 8F 89 0616 1167 BISB3 #^X80,R3,DHUTBF2(R0) ; load high address and start
05 061C 1168 RSB
```



```
00000200 8F 52 D1 061D 1170 : Code to do mapped DMA transfers
      52 0200 8F 3C 061D 1171 : DMA_MAP:
      011C C5 52 C0 061D 1172 :
      0120 C5 52 A2 061D 1173 :
      21 0134 C5 02 E0 061D 1174 : CMPL R2,#512 ; NEXT BURST TOO LONG FOR MAPS?
      0624 1175 : BLEQ 5$ ; NO
      0626 1176 : MOVZWL #512,R2
      062B 1177 :
      062B 1178 :
      011C C5 52 C0 062B 1179 5$: ADDL R2,UCB$$_TT_OUTADR(R5) ; UPDATE CHARACTER POINTER FOR NEXT TIME
      0120 C5 52 A2 0630 1180 : SUBW R2,UCB$$_TT_OUTLEN(R5) ; UPDATE COUNT FOR NEXT TIME
      21 0134 C5 02 E0 0635 1181 :
      0635 1182 : BBS #UCB$$_XOFF,UCB$$_DHUFLG(R5),6$ ; IF XOFF, DON'T SET TIMER
      063B 1183 : TIMSET R2,R1,LOCKOUTPUT ; RECOMPUTE TIMEOUT VALUE FOR THIS
      065C 1184 : ; PORTION OF THE DMA BURST
      065C 1185 :
      065C 1186 :
      065C 1187 : R3 - STRING ADDRESS
      065C 1188 : R2 - LENGTH
      065C 1189 : R5 - UCB
      065C 1190 :
      50 24 BB 065C 1191 6$: PUSH R2,R5>
      50 24 A5 D0 065E 1192 : MOV L UCB$$_CRB(R5),R0 ; GET CRB ADDRESS
      51 38 B0 D0 0662 1193 : MOV L @CRB$$_INTD+VEC$$_ADP(R0),R1 ; CONFIG REGISTER
      50 0F 00 EF 0666 1194 : EXTZV #VEC$$_MAPREG,#VEC$$_MAPREG,-
      50 012C C5 0669 1195 : MOV L UCB$$_TP_MAP(R5),R0 ; GET STARTING MAP REGISTER
      51 0800 C140 DE 066D 1196 : MOVAL UCB$$_MAP(R1)[R0],R1 ; GET 1ST MAP REGISTER ADDRESS
      54 53 80000000 8F CB 0673 1197 :
      54 54 F7 8F 78 067B 1198 : BICL3 #X80000000,R3,R4 ; CALC SVAPTE OF BUFFER
      55 00000000 GF D0 0680 1200 : ASHL #9,R4,R4 ; ISOLATE PAGE
      54 6544 DE 0687 1201 : MOV L G^MMG$$_SPTBASE,R5 ; GETS SVAPTE OF BUFFER
      53 FFFFE00 8F CA 068B 1202 : MOVAL (R5)[R4],R4 ; INTO R4
      0692 1203 : BICL #C^X1FF,R3 ; COMPUTE BYTE OFFSET IN PAGE
      0692 1204 :
      0692 1205 : LOAD MAP REGISTERS
      0692 1206 : R0 - MAP REGISTER NUMBER
      0692 1207 : R1 - ADDRESS OF FIRST MAP REGISTER
      0692 1208 : R2 - BUFFER LENGTH
      0692 1209 : R3 - BYTE OFFSET IN PAGE
      0692 1210 : R4 - SVAPTE OF BUFFER
      52 02 9A 0692 1211 : MOVZBL #2,R2
      55 84 D0 0695 1212 10$: MOVL (R4)+,R5 ; GET CONTENTS OF NEXT PTE
      0698 1213 :
      0698 1214 : THIS CODE ASSUMES THAT DMA IS FROM NONPAGED POOL
      0698 1215 :
      55 0B 15 00000400 8F F0 0698 1216 : INSV #X400,#21,#11,R5 ; SET VALID BIT, DATA PATH 0
      81 55 D0 06A1 1217 : MOVL R5,(R1)+ ; LOAD INTO MAP REGISTER
      EE 52 F5 06A4 1218 : SOBGR R2,10$
      24 BA 06A7 1219 :
      06A7 1220 : POPR #M<R2,R5> ; RESTORE LENGTH,WRITE BUFFER, UCB
      06A9 1221 :
      06A9 1222 :
      06A9 1223 : Note that the following code works with 22 bit Qbus addresses
      06A9 1224 : As well as with 18 bit Unibus addresses
      53 0D 09 50 F0 06A9 1225 :
      06A9 1226 : INSV R0,#9,#13,R3 ; COMPUTE UNIBUS ADDRESS
```



```

53 53 53 80 8F 89 8E 05 06AE 1227 SETIND
OC AO 53 06 10 EF 89 06BD 1228 MOVW R2,DHUTCT(R0) ; Load count
OA AO 53 06 10 EF 89 06C1 1229 MOVW R3,DHUTBF1(R0) ; Load low bits
53 53 53 80 8F 89 06C5 1230 EXTZV #16,#6,R3,R3 ; get high bits
OC AO 53 06 10 EF 89 06CA 1231 BISB3 #^X80,R3,DHUTBF2(R0) ; load high bits and initiate DMA
53 53 53 80 8F 89 06D0 1232 POPL R0 ; RESTORE CSR ADDRESS
OC AO 53 06 10 EF 89 06D3 1233 RSB ; RETURN TO CALLER
OA AO 53 06 10 EF 89 06D4 1234 ; FORK DISPATCHER, ISR, OR STARTIO
53 53 53 80 8F 89 06D4 1235 DMA_DONE: ; DMA COMPLETION
OC AO 53 06 10 EF 89 06D4 1236
OA AO 53 06 10 EF 89 06D4 1237
53 53 53 80 8F 89 06D4 1238
OC AO 53 06 10 EF 89 06D4 1239 BBS #TTY$V_PC_PRMMAP,- ; SKIP FORK IF MAPS PERMANENT
OA AO 53 06 10 EF 89 06D6 1240 UCBSW TT_PRTCTL(R5),DMA_POST ; SHOW DEALLOC FORK ACTIVE
53 53 53 80 8F 89 06DA 1241 BISB #TTY$M_TP_DLLOC,-
OC AO 53 06 10 EF 89 06DC 1242 UCBSB_TP_STAT(R5)
OA AO 53 06 10 EF 89 06DF 1243 MOVL UCBSL-TT-CLASS(R5),R1 ; GET CLASS VECTOR ADDRESS
53 53 53 80 8F 89 06E4 1244 JSB @CLASS_FORK(R1) ; SCHEDULE FORK TO FIPL FOR MAP
OC AO 53 06 10 EF 89 06E7 1245 RSB ; REGISTER DEALLOCATION
OA AO 53 06 10 EF 89 06E7 1246 ; RETURN TO CALLER, FORK WILL RESUME
53 53 53 80 8F 89 06E8 1247 ; AT DMA_DEALLOC
OC AO 53 06 10 EF 89 06E8 1248
OA AO 53 06 10 EF 89 06E8 1249 DMA_DEALLOC:
53 53 53 80 8F 89 06E8 1250 MOVL UCBSL-CRB(R5),R0 ; GET CRB ADDRESS
OC AO 53 06 10 EF 89 06EC 1251 MOVL UCBSL-TP_MAP(R5),-
OA AO 53 06 10 EF 89 06F0 1252 CRBSL-INTD+VECSW_MAPREG(R0); RESTORE MAP FIELD IN CRB
53 53 53 80 8F 89 06F2 1253 BEQL $$ ; SKIP IF NONE
OC AO 53 06 10 EF 89 06F4 1254 JSB G^IOCS$RELMAPREG ; RELEASE MAP REGISTERS
OA AO 53 06 10 EF 89 06FA 1255
53 53 53 80 8F 89 06FA 1256 $$: SETIPL UCBSB_DIPL(R5) ; INTERLOCK TO DEVICE IPL
OC AO 53 06 10 EF 89 06FE 1257 BICW #TTY$M_PC_MAPAVL,- ; SHOW MAP ALLOCATED
OA AO 53 06 10 EF 89 0700 1258 UCBSW TT_PRTCTL(R5)
53 53 53 80 8F 89 0703 1259 BICB #TTY$M_TP_DLLOC,- ; SHOW DEALLOC FORK DONE
OC AO 53 06 10 EF 89 0705 1260 UCBSB_TP_STAT(R5)
OA AO 53 06 10 EF 89 0708 1261
53 53 53 80 8F 89 0708 1262 DMA_POST:
OC AO 53 06 10 EF 89 0708 1263 BICB #TTY$M_TP_ABORT,- ; RESET ABORT REQUEST
OA AO 53 06 10 EF 89 070A 1264 UCBSB_TP_STAT(R5)
53 53 53 80 8F 89 070D 1265 BICW #TTY$M_TANK_DMA,- ; RESET DMA MODE
OC AO 53 06 10 EF 89 0711 1266 UCBSW-TT_HOLD(R5)
OA AO 53 06 10 EF 89 0714 1267
53 53 53 80 8F 89 0714 1268 ; CALL GETNEXT TO CONTINUE PROCESSING
OC AO 53 06 10 EF 89 0714 1269
OA AO 53 06 10 EF 89 0714 1270 BICB #UCBSM_TIM!UCBSM_INT,- ; CLEAR TIMEOUT AND INT EXPECTED
53 53 53 80 8F 89 0716 1271 UCBSW_STS(R5)
OC AO 53 06 10 EF 89 0718 1272 JSB @UCBSL-TT_GETNXT(R5) ; GET NEXT BURST
OA AO 53 06 10 EF 89 071C 1273 BRW YF$STARTIO ; AND PROCEED
53 53 53 80 8F 89 071F 1274 YF$FORK:
OC AO 53 06 10 EF 89 071F 1275 SAVIPL ; SAVE CURRENT IPL ON THE STACK
OA AO 53 06 10 EF 89 0722 1276 PUSHAL 20$ ; BUILD RETURN ADDRESS ON STACK
53 53 53 80 8F 89 0728 1277 BBC #TTY$V_TP_ALLOC,- ; SKIP IF NOT ALLOCATE FORK
OC AO 53 06 10 EF 89 072A 1278 UCBSB_TP_STAT(R5),10$
OA AO 53 06 10 EF 89 072E 1279 BRW DMA_ACLOC ; RESUME AT ALLOCATE CODE THREAD
53 53 53 80 8F 89 0731 1280 10$:
OC AO 53 06 10 EF 89 0731 1281 BBC #TTY$V_TP_DLLOC,- ; CHECK FOR DEALLOCATE
OA AO 53 06 10 EF 89 0733 1282 UCBSB_TP_STAT(R5),20$
53 53 53 80 8F 89 0737 1283 BRW DMA_DEALLOC
```


- Port Driver for DHU/DHV
PORT DMA ROUTINES

```
16-SEP-1984 02:26:48 VAX/VMS Macro V04-00
5-SEP-1984 04:17:43 [TTDRVR.SRC]YFDRIVER.MAR;1
```

Page 30
(1)

```
; RESTORE SAVED FORK IPL FROM STACK
```

YFDF
Symt

[illegible]


```
073E 1290 .SBTTL PORT ROUTINES STOP,RESUME,XON,XOFF
073E 1291 :++
073E 1292 : YF$XOFF - SEND XOFF
073E 1293 : YF$XON - SEND XON
073E 1294 : YF$STOP - STOP OUTPUT
073E 1295 : YF$ABORT - ABORT CURRENT OUTPUT
073E 1296 : YF$RESUME - RESUME STOPPED OUTPUT
073E 1297 :
073E 1298 : FUNCTIONAL DESCRIPTION:
073E 1299 :
073E 1300 : THESE ROUTINES ARE USED BY THE THE TERMINAL CLASS DRIVER TO
073E 1301 : CONTROL OUTPUT ON THE PORT
073E 1302 :
073E 1303 : INPUTS:
073E 1304 :
073E 1305 : R5 = UCB ADDRESS
073E 1306 :
073E 1307 : OUTPUTS:
073E 1308 :
073E 1309 : R5 = UCB ADDRESS
073E 1310 : --
073E 1311 :
073E 1312 : SCHEDULE XOFF OR XON TO BE SEND
073E 1313 :
073E 1314 : INPUTS:
073E 1315 :
073E 1316 : R3 - CONTAINS THE CHARACTER TO SEND AS FLOW CONTROL.
073E 1317 :
073E 1318 :
073E 1319 : To send an XON we just clear the Force XOFF bit
073E 1320 YF$XON:
073E 1321 :
073E 1322 : Forget any stored 'XOFF' character
0100 8F AA 073E 1323 BICW #TTY$M_TANK_PREMPT,- ; RESET XOFF STATE
0108 C5 0742 1324 UCBSW_TT_HOLD(R5)
0745 1325 SETIND
0754 1326 :
0754 1327 : Clearing this bit will make the device send an XON asap
0754 1328 :
08 A0 20 AA 0754 1329 BICW #DHULCT$M_SNDOFF,DHULCT(R0)
11 53 91 0758 1330 CMPB R3,#^X11 ; Is it XOFF ?
1A 12 075B 1331 BNEQ YF$PREEMPT
05 075D 1332 RSB
075E 1333 :
075E 1334 : To send an XOFF we just set the Force XOFF bit,
075E 1335 :
075E 1336 YF$XOFF:
075E 1337 SETIND
13 53 91 076D 1338 CMPB R3,#^X13 ; Is it XOFF ?
08 A0 05 12 0770 1339 BNEQ YF$PREEMPT ; Not XOFF, have to do it the hard way
20 A8 0772 1340 BISW #DHULCT$M_SNDOFF,DHULCT(R0)
05 0776 1341 RSB
0777 1342 :
0777 1343 : we have to send a character here (other than normal XON/XOFF),
0777 1344 : so see if the device is idle
0777 1345 :
0777 1346 YF$PREEMPT:
```



```
3B 64 A5 01 E2 0777 1347 BBSS #UCBS$V_INT,UCBS$W_STS(R5),30$ ; Branch if active
OD A0 95 077C 1348 TSTB DHUTBF2+1(R0)
21 18 077F 1349 BGEQ 10$ ; XOFFED, don't set timer
0781 1350 TIMSET #1,R1,LOCKOUTPUT
07A2 1351
07A2 1352 : If this is a DHV then we send the char by single character,
07A2 1353 : else by fifo
07A2 1354
07A2 1355 10$:
02 A0 09 0134 C5 01 E0 07A2 1356 BBS #UCBS$V_DHU,UCBS$B_DHUFLG(R5),20$
53 8000 8F A9 07A8 1357 BISW3 #^X8000,R3,DHUTXC(R0)
12 11 07AF 1358 BRB 90$
06 A0 53 90 07B1 1359 20$:
OC 11 07B1 1360 MOVB R3,DHUTXF(R0)
07B5 1361 BRB 90$
07B7 1362
07B7 1363 : Transmission is in progress, save the character till
07B7 1364 : the transmission completes
07B7 1365 30$:
0100 8F A8 07B7 1366 BISW #TTY$M_TANK_PREMPT,- ; Set the flag
0108 C5 07BB 1367 UCBS$W_TP_HOLD(R5)
010A C5 53 90 07BE 1368 MOVB R3,UCBS$B_TP_PREMPT(R5) ; Save the character
07C3 1369
07C3 1370 90$:
05 07C3 1371 RSB
07C4 1372
07C4 1373 : STOP PORT OUTPUT
07C4 1374
07C4 1375 : YF$STOP:
07C4 1376
50 DD 07C4 1377 PUSHL R0
07C6 1378 SETIND
OD A0 94 07D5 1379 CLRB DHUTBF2+1(R0)
07D8 1380
07D8 1381 : Note. We dont reset UCBS$M_INT for the DHU,DHV in case the
07D8 1382 : device finishes transmitting
07D8 1383
0134 C5 04 88 07D8 1384 BISB #UCBS$M_XOFF,UCBS$B_DHUFLG(R5) ; set to indicate xoff
64 A5 01 8A 07DD 1385 BICB #UCBS$M_TIM,UCBS$W_STS(R5) ; Reset timer
01 01 BA 07E1 1386 POPR #^M<R0>
05 07E3 1387 RSB
07E4 1388
07E4 1389 : ABORT ANY CURRENT PORT OUTPUT ACTIVITY
07E4 1390
07E4 1391 YF$ABORT:
18 64 A5 50 DD 07E4 1392 PUSHL R0
0130 C5 01 E1 07E6 1393 BBC #UCBS$V_INT,UCBS$W_STS(R5),15$ ; SKIP IF NOT BUSY.
01 88 07EB 1394 BISB #TTY$M_TP_ABORT,UCBS$B_TP_STAT(R5) ; REQUEST DMA ABORT
07F0 1395 SETIND
08 A0 01 A8 07FF 1396 BISW #DHULCT$M_ABORT,DHULCT(R0)
01 01 BA 0803 1397 15$:
05 0803 1398 POPR #^M<R0>
0805 1399 RSB
0806 1400
0806 1401 : RESUME PREVIOUSLY STOPPED PORT OUTPUT
0806 1402
0806 1403 YF$RESUME:
```



```

      OF  BB  0806  1404      PUSH  #^M<R0,R1,R2,R3>
      0808  1405      SETIND
OD A0  80  8F  90  0817  1406      MOV  #^X80,DHUTBF2+1(R0)      ; ENABLE TRANSMIT
      0134 C5  04  8A  081C  1407      BICB  #UCBSM_XOFF,UCBSB_DHUTLG(R5)      ; clear to indicate xon
      33 64 A5  01  E0  0821  1408      BBS  #UCBSV_INT,UCBSW_STS(R5),50$      ; Recalculate timeout
2A 0108 C5  0B  E5  0826  1409      BBCC  #TTY$V_TANK_BURST,UCBSW_IT_HOLD(R5),40$      ; NO BURST IN PROGRESS
      082C  1410      ; (RESET ANYWAY. WILL BE
      082C  1411      ; SET IF NEEDED BY BURST_OUT
      51  0120 C5  3C  082C  1412      MOVZWL UCBSW_IT_OUTLEN(R5),R1      ; GET NUMBER CHARACTERS
      0831  1413      TIMSET R1,R1,LOCKOUTPUT      ; COMPUTE TIMEOUT AND
      0852  1414      ; SET INTERRUPT EXPECTED
      FBED CF  16  0852  1415      JSB  BURST_OUTPUT      ; RESTART OUTPUT
      0856  1416 40$:      POPR  #^M<R0,R1,R2,R3>
      OF  BA  0856  1417      RSB
      05  0858  1418
      0859  1419
      0859  1420 50$:
      0859  1421      ;
      0859  1422      ; reset timeout for the DHU/DHV
      0859  1423      ;
      0859  1424      ; We use the number of characters that the DHU has got left to send
      0859  1425      ; plus the number in the remaining part of the buffer
      0859  1426      ; Note that the byte count may not be updated by the DHU/DHV
      0859  1427      ; so we will over estimate the timeout value
      0859  1428      ; We have to add 64 because these characters may be in the Transmit fifo
      51  0040 8F  3C  0859  1429      MOVZWL #64,R1
      51  51  0E A0  A0  085E  1430      ADDW  DHUTCT(R0),R1
      51  0120 C5  A0  0862  1431      ADDW  UCBSW_IT_OUTLEN(R5),R1
      0867  1432      TIMSET R1,R1,LOCKOUTPUT      ; COMPUTE TIMEOUT AND
      0888  1433      ; SET INTERRUPT EXPECTED
      CC  11  0888  1434      BRB  40$
      088A  1435
```



```
088A 1437 .SBTTL OUTPUT INTERRUPT SERVICE
088A 1438 :++
088A 1439 :YF$INTOUT - DHU OUTPUT INTERRUPT SERVICE
088A 1440 :
088A 1441 :FUNCTIONAL DESCRIPTION:
088A 1442 :
088A 1443 :THIS ROUTINE IS ENTERED WHEN THE DHU FINDS A LINE ENABLED
088A 1444 :AND AN EMPTY UART. THE CORRESPONDING UCB IS FOUND AND
088A 1445 :ANY OUTSTANDING PORT OUTPUT IS DONE. WHEN ALL OUTSTANDING PORT
088A 1446 :OUTPUT IS COMPLETED, THE CLASS DRIVER IS CALLED TO RETURN THE NEXT
088A 1447 :CHARACTER OR STRING TO BE OUTPUT. IF NO MORE OUTPUT IS FOUND, THEN
088A 1448 :THE LINE IS DISABLED.
088A 1449 :
088A 1450 :INPUTS:
088A 1451 :
088A 1452 :SP(00) = ADDRESS OF THE IDB
088A 1453 :
088A 1454 :IMPLICIT INPUTS:
088A 1455 :
088A 1456 :R0,R1,R2,R3,R4,R5 SAVED ON THE STACK.
088A 1457 :
088A 1458 :OUTPUTS:
088A 1459 :
088A 1460 :THE INTERRUPT IS DISMISSED.
088A 1461 :
088A 1462 :--
088A 1463 YF_OUT_EXIT:
088A 1464 ADDL #4,SP ; EXIT OUTPUT INTERRUPT
088D 1465 MOVQ (SP)+,R0 ; REMOVE IDB ADDRESS
0890 1466 MOVQ (SP)+,R2 ; RESTORE REGISTERS
0893 1467 MOVQ (SP)+,R4 ;
0896 1468 REI ; DISMISS INTERRUPT
0897 1469
0897 1470 YF$INTOUT:: ; DHU OUTPUT INTERRUPT SERVICE
0897 1471
0897 1472 YF_OUT_LOOP:
0897 1473 MOVL @ (SP),R4 ; GET THE IDB ADDRESS
089B 1474 MOVL (R4),R0 ; GET THE CSR ADDRESS
089E 1475 :
089E 1476 :GET THE LINE INFO FROM THE CSR
089E 1477 :
089E 1478
089E 1479 MOVW (R0),R2 ; GET THE CSR VALUE
08A1 1480 BGEQ YF_OUT_EXIT
08A3 1481 ASHL #-2,R2,R1 ; Get the line number
08A8 1482 BICL #^C<15>,R1
08AF 1483 MOVL IDB$U_CBLST(R4)(R1),R5 ; GET THE UCB ADDRESS
08B4 1484 BEQL YF_OUT_LOOP ; IF EQL THEN DISMISS
08B6 1485 :
08B6 1486 :CHECK FOR BURST OR DMA ACTIVE ON LINE
08B6 1487 :
08B6 1488 SETIND R0
08BD 1489 BITW #DHULCT$M_ABORT, DHULCT(R0) ; CHECK TO SEE IF ABORT IS SET
08C1 1490 BEQL 5$ ; EQUAL, ABORT NOT SET
08C3 1491 BICW #DHULCT$M_ABORT, DHULCT(R0) ; CLEAR ABORT
08C7 1492 5$: CMPB #TTY$M_TANK_BURST-8,- ; ONLY BURST ACTIVE?
08C9 1493 UCBSW_TT_HOLD+1(R5)
```

```

      42 13 08CC 1494      BEQL  YF_SILO      ; YES, CONTINUE SILO OUTPUT
0100 8F B3 08CE 1495      BITW  #TTY$M_TANK_PREMPT,- ; Preempt required ?
0108 C5      08D2 1496      UCB$W_TT_HOLD(R5)
      2C 12 08D5 1497      BNEQ  40$          ; == BRW YF_PREEMPT
      08D7 1498 9$:
1000 8F B3 08D7 1499      BITW  #TTY$M_TANK_DMA,- ; DMA ACTIVE?
0108 C5      08DB 1500      UCB$W_TT_HOLD(R5)
      26 12 08DE 1501      BNEQ  45$          ; YES, PROCESS IT. (== BRW YF_DMA_INTERRUPT)
      08E0 1502 : NO PENDING DATA - LOOK FOR NEXT CHARACTER
      08E0 1503 :
      08E0 1504 :
64 A5 03 8A 08E0 1505 10$: BICB  #UCB$M_TIM!UCB$M_INT,UCB$W_STS(R5); CLEAR TIMEOUT AND EXPECTED
      08E4 1506 :
      08E4 1507 : CALL CLASS DRIVER FOR MORE OUTPUT
      08E4 1508 :
010C D5 16 08E4 1509      JSB    @UCB$L_TT_GETNXT(R5) ; GET THE NEXT CHARACTER
      1F 19 08E8 1510      BLSS  YF_START_BURST ; BURST SPECIFIED
      AB 13 08EA 1511      BEQL  YF_OUT_LOOP ; NONE
      08EC 1512 :
      08EC 1513 : OUTPUT A CHARACTER TO THE DHU/DHV
      08EC 1514 :
      08EC 1515 20$:
0A 0134 C5 01 E0 08EC 1516      BBS    #UCB$V_DHU,UCB$B_DHUFLG(R5),30$
      08F2 1517 :
      08F2 1518 : Single char output to DHV
      08F2 1519 :
02 A0 53 8000 8F A9 08F2 1520      BISW3 #^X8000,R3,DHUTXC(R0)
      FF9B 31 08F9 1521      BRW    YF_OUT_LOOP
      08FC 1522 :
      08FC 1523 : Fifo output to DHU
      08FC 1524 :
      08FC 1525 30$:
06 A0 53 90 08FC 1526      MOVB  R3,DHUTXF(R0)
      FF94 31 0900 1527 35$:
      0900 1528      BRW    YF_OUT_LOOP
      0089 31 0903 1529 40$:
      0903 1530      BRW    YF_PREEMPT
      00A8 31 0906 1531 45$:
      0906 1532      BRW    YF_DMA_INTERRUPT
      0909 1533
      0909 1534
```



```
0909 1536
FB36 CF 16 0909 1537 YF_START BURST:
FF87 31 0909 1538 JSB BURST_OUTPUT ; START OUTPUT SILO OR DMA
090D 1539 BRW YF_OUT_LOOP
0910 1540 :
0910 1541 : CONTINUE SILO OUTPUT
0910 1542 :
0910 1543 YF_SILO:
0910 1544 :
0910 1545 : Note. THE DHV does not have SILO output
0910 1546 :
09 0134 C5 01 E1 0910 1547 BBC #UCBSV_DHU,UCBSB_DHUFLG(R5),5$
40 8F 06 A0 91 0916 1548 CMPB DHUTFSTR0),#64 ; ANY SILO OUTPUT IN PROGRESS ?
22 12 091B 1549 BNEQ 15$ ; YES THEN LET IT COMPLETE
26 11 091D 1550 BRB 20$ ; OTHERWISE CONTINUE SILO OUTPUT
091F 1551 :
091F 1552 : DHV single char mode
091F 1553 :
091F 1554 5$:
02 A0 53 011C D5 9A 091F 1555 MOVZBL @UCBSL_TT_OUTADR(R5),R3
53 8000 8F A9 0924 1556 BISW3 #^X8000,R3,DHUTXC(R0)
011C C5 D6 092B 1557 INCL UCBSL_TT_OUTADR(R5)
0120 C5 B7 092F 1558 DECW UCBSW_TT_OUTLEN(R5)
07 12 0933 1559 BNEQ 10$ ; NOT DONE
0800 8F AA 0935 1560 BICW #TTY$M_TANK_BURST,- ; RESET BURST ACTIVE
0108 C5 0939 1561 UCBSW_TT_HOCD(R5)
FF58 31 093C 1562 10$: BRW YF_OUT_LOOP
093F 1564 15$:
F6F5 CF D6 093F 1566 INCL YFSL_SIL_ERROR ; INCREMENT ERROR COUNTER
F7 11 0943 1567 BRB 10$ ; == BRW YF_OUT_LOOP
0945 1568 :
0945 1569 : DHU silo output
0945 1570 :
0945 1571 20$:
51 06 A0 9A 0945 1572 MOVZBL DHUTFS(R0),R1 ; number of slots
0949 1573 :
52 0120 C5 3C 0949 1574 MOVZWL UCBSW_TT_OUTLEN(R5),R2 ; GET CURRENT LENGTH
53 011C C5 D0 094E 1575 MOVL UCBSL_TT_OUTADR(R5),R3 ; GET CURRENT ADDRESS
51 52 B1 0953 1576 CMPW R2,R1 ; BURST LARGER THAN SILO?
03 1B 0956 1577 BLEQU 50$ ; NO
52 51 9A 0958 1578 MOVZBL R1,R2 ; MAXIMUM
095B 1579 50$:
011C C5 52 C0 095B 1580 ADDL R2,UCBSL_TT_OUTADR(R5) ; UPDATE POINTER
0120 C5 52 A2 0960 1581 SUBW R2,UCBSW_TT_OUTLEN(R5) ; AND COUNT
07 12 0965 1582 BNEQ 60$ ; NOT DONE
0800 8F AA 0967 1583 BICW #TTY$M_TANK_BURST,- ; RESET BURST ACTIVE
0108 C5 096B 1584 UCBSW_TT_HOCD(R5)
52 D5 096E 1585 60$: TSTL R2 ; ANY ROOM AT ALL
1A 13 0970 1586 BEQL 80$ ; NO THEN EXIT
0972 1587 :
06 A0 08 52 E9 0972 1588 BLBC R2,70$ ; EVEN TRANSFER
83 90 0975 1589 MOVW (R3)+,DHUTXF(R0) ; OUTPUT ODD BYTE
52 D7 0979 1590 DECL R2 ; UPDATE COUNT
0F 13 097B 1591 BEQL 80$ ; DONE
097D 1592 70$:
```



```
52 52 FF 8F 78 097D 1593 ASHL # -1,R2,R2 ; CONVERT TO WORD COUNT
06 A0 83 B0 0982 1594 75$: MOVW (R3)+,DHUTXF(R0)
F6 52 F5 0986 1596 DELAY ; TO RELEASE THE UNIBUS
FF08 31 0989 1597 SOBGTR R2,75$ ; LOOP TILL DONE
098C 1598
098F 1599 80$: BRW YF_OUT_LOOP
098F 1600 YF_PREEMPT:
0108 C5 0100 8F AA 098F 1602 BICW #TTY$M_TANK_PREMPT,UCB$W_TT_HOLD(R5) ; CLEAR PREEMPT BIT
53 010A C5 9A 0996 1603 MOVZBL UCB$B_TT_PREMPT(R5),R3 ; -GET CHARACTER
09 0134 C5 01 E0 0998 1604 BBS #UCB$V_DHU,UCB$B_DHUFLG(R5),10$
09A1 1605 ;
09A1 1606 ; DHV single character
02 A0 53 8000 8F A9 09A1 1607 BISW3 #^X8000,R3,DHUTXC(R0)
04 11 09A8 1608 BRB 20$
09AA 1609 ;
09AA 1610 ; DHU FIFO output
09AA 1611
09AA 1612 10$:
06 A0 53 90 09AA 1613 MOVB R3,DHUTXF(R0)
FEE6 31 09AE 1614 20$:
09AE 1615 BRW YF_OUT_LOOP
09B1 1616
09B1 1617 YF_DMA_INTERRUPT:
09B1 1618 ;
09B1 1619 ; CHECK TO MAKE SURE NO DATA IS PENDING BEFORE ASKING FOR MORE
09B1 1620 ;
04 FEE2 CF DF 09B1 1621 PUSHAL YF_OUT_LOOP ; BUILD RETURN ADDRESS ON STACK
52 0C E1 09B5 1622 BBC #DHUCSR$V_DMAERR,R2,$$ ; CHECK FOR A DMA ERROR
F683 CF D6 09B9 1623 INCL YF$L_DMA$MT_ERROR ; ERROR OCCURED INCREMENT COUNTS
09BD 1624 5$:
09BD 1625 BITB #TTY$M_TP_ALLOC!TTY$M_TP_DLLOC,- ;CHECK FOR FORKS ACTIVE
0130 C5 06 93 09BF 1626 UCB$B_TP_STAT(R5) ; AND IGNORE IF SO
1C 12 09C2 1627 BNEQ 20$
0130 C5 00 E0 09C4 1628 BBS #TTY$V_TP_ABORT,UCB$B_TP_STAT(R5),-
10 09C9 1629 10$ ; ABORT ACTIVE DMA
0C A0 95 09CA 1630 TSTB DHUTBF2(R0) ; ANY DMA IN PROGRESS ?
12 19 09CD 1631 BLSS 30$ ; YES, then let it complete
0E A0 B5 09CF 1632 TSTW DHUTCT(R0) ; TEST DMA BYTE COUNT
06 13 09D2 1633 BEQL 10$ ; DMA BYTE COUNT DONE
F664 CF D6 09D4 1634 INCL YF$L_ERROR
06 11 09D8 1635 BRB 20$ ; NO THEN CONTINUE
09DA 1636 10$:
0E A0 B4 09DA 1637 CLRW DHUTCT(R0) ; CLEAR DMA BYTE COUNT (ABORT WAS CLEARED)
FB7D 31 09DD 1638 BRW DMA_CONTINUE ; OTHERWISE, CONTINUE THE DMA
09E0 1639
09E0 1640 ; IF THIS INTERRUPT WAS THE RESULT
09E0 1641 ; OF AN ABORT, THIS WILL BE HANDLED
09E0 1642 ; BY DMA_CONTINUE
05 09E0 1643 20$:
09E0 1644 RSB
09E1 1645
09E1 1646
09E1 1647 30$:
F653 CF D6 09E1 1648 INCL YF$L_SIL_ERROR
5E 04 C0 09E5 1649 ADDL #4,SP ; Pop off return address
```


YFDRIVER
V04-000

- Port Driver for DHU/DHV
OUTPUT INTERRUPT SERVICE

J 8

16-SEP-1984 02:26:48
5-SEP-1984 04:17:43

VAX/VMS Macro V04-00
[TTDRVR.SRC]YFDRIVER.MAR;1

Page 38
(1)

FEAC 31 09EB 1650 BRW YF_OUT_LOOP
09EB 1651

```
09EB 1653 .SBTTL SET SPEED, PARITY PARAMETERS
09EB 1654
09EB 1655 :++
09EB 1656 : YFSSET_LINE - RESET SPEED, PARITY
09EB 1657 :
09EB 1658 : FUNCTIONAL DESCRIPTION:
09EB 1659 :
09EB 1660 : INPUTS:
09EB 1661 :
09EB 1662 : R5 - UCB ADDRESS
09EB 1663 :
09EB 1664 : OUTPUTS:
09EB 1665 :
09EB 1666 : R4 USED
09EB 1667 :--
09EB 1668
09EB 1669 YFSSET_LINE:
54 24 0C BB 09EB 1670 PUSHR #^M<R2,R3>
54 2C B4 DO 09ED 1671 MOVL UCB$$_CRB(R5),R4 ; ADDRESS CRB
52 08 A0 3C 09F1 1672 MOVL @CRB$$_INTD+VEC$_IDB(R4),R4 ; GET THE CSR ADDRESS VIA CRB
09F5 1673 SETIND
0A04 1674 MOVZWL DHULCT(R0),R2 ; Fetch the line control reg
0A08 1675
0A08 1676 :
0A08 1677 : The DHU/DHV have automatic detection of received XON/XOFF and also
0A08 1678 : automatic generation of XON/XOFF options
0A08 1679 :
14 0122 C5 05 E1 0A08 1680 BBC #TTY$_PC_XOFAVL,UCB$_TT_PRTCTL(R5),4$; AUTOXON XOFF AVAILABLE ON T
0A0E 1681 ; YES THEN IS
0A0E 1682 :
0A0E 1683 : Assume that both modes are required
0A0E 1684 :
12 A8 0A0E 1685 BISW #<DHULCT$_OAUTO ! DHULCT$_IAUTO>,-
52 0A10 1686 R2; Assume AUTOXOFF
0A11 1687 :
0A11 1688 : Disable detection of received XON/XOFF if not allowed
0A11 1689 :
03 0122 C5 06 E0 0A11 1690 BBS #TTY$_PC_XOFENA,UCB$_TT_PRTCTL(R5),2$; AUTOXON XOFF ENABLED
52 02 AA 0A17 1691 BICW #DHULCT$_OAUTO,R2 ; NO THEN CLEAR THE AUTOXOFF ENABLE
0A1A 1692 2$:
0A1A 1693 :
0A1A 1694 : Disable sending of XON/XOFF is hostsync is not set
0A1A 1695 : (The DHU/DHV can send XON/XOFF automatically if the receive fifo fills up)
0A1A 1696 :
03 44 A5 04 E0 0A1A 1697 BBS #TTY$_HOSTSYNC,UCB$_DEVDEPEND(R5),4$; Host sync specified ?
52 10 AA 0A1F 1698 BICW #DHULCT$_IAUTO,R2 ; No, then clear enable flag
0A22 1699 :
52 0100 8F AA 0A22 1700 4$: BICW #DHULCT$_MODEM,R2
05 44 A5 15 E1 0A27 1701 BBC #TTY$_MODEM,UCB$_DEVDEPEND(R5),6$
52 0100 8F A8 0A2C 1702 BISW #DHULCT$_MODEM,R2
0A31 1703 :
0A31 1704 :
0A31 1705 : move updated register back into device
0A31 1706 :
08 A0 52 B0 0A31 1707 6$: MOVW R2,DHULCT(R0)
7E D4 0A35 1708 CLRL -(SP) ; RESET A TEMPORARY LOCATION
0A37 1709 :
```



```
0A37 1710 : SET UP LINE SPEED AND PARITY
0A37 1711 :
0A37 1712 :
00F5 C5 95 0A37 1713 TSTB UCBSW_TT_SPEED+1(R5) ; RECEIVE SPEED SPECIFIED?
07 12 0A3B 1714 BNEQ 8$ ; YES
00F4 C5 90 0A3D 1715 MOVB UCBSW_TT_SPEED(R5) ; NO, SO USE TRANSMITTER SPEED
00F5 C5 0A41 1716 UCBSW_TT_SPEED+1(R5)
53 00F4 C5 9A 0A44 1717 8$: MOVZBL UCBSW_TT_SPEED(R5),R3 ; Get TRANSMIT SPEED
53 F637 CF43 90 0A49 1719 MOVB YF$VMS_SPEEDS[R3],R3
19 19 0A4F 1720 BLSS 10$ ; Illegal speed
6E 04 0C 53 F0 0A51 1721 INSV R3,#DHULPR$V_TSPEED,#4,(SP) ; SET TRANSMIT SPEED
53 00F5 C5 9A 0A56 1722 MOVZBL UCBSW_TT_SPEED+1(R5),R3
53 F625 CF43 90 0A5B 1723 MOVB YF$VMS_SPEEDS[R3],R3
07 19 0A61 1724 BLSS 10$ ; Illegal speed
6E 04 08 53 F0 0A63 1725 INSV R3,#DHULPR$V_RSPEED,#4,(SP) ; SET RECEIVE SPEED
27 11 0A68 1726 BRB 20$ ; set current speed
0A6A 1727
0A6A 1728 10$:
0A6A 1729 :
0A6A 1730 : This code restores the speed to its previous value
0A6A 1731 : It is entered when an illegal speed combination is detected.
0A6A 1732
53 04 A0 B0 0A6A 1733 MOVW DHULPR(R0),R3 ; get line parameters
6E 53 B0 0A6E 1734 MOVW R3,(SP) ; use previous speed as new
6E 94 0A71 1735 CLRB (SP)
53 53 04 0C EF 0A73 1736 EXTZV #DHULPR$V_TSPEED,#4,R3,R3 ; extract speed
00F4 C5 F618 CF43 90 0A78 1737 MOVB YF$DHU_SPEEDS[R3],UCBSW_TT_SPEED(R5); convert to VMS value
0A80 1738
53 53 04 A0 B0 0A80 1739 MOVW DHULPR(R0),R3 ; get line parameters
53 53 04 08 EF 0A84 1740 EXTZV #DHULPR$V_RSPEED,#4,R3,R3 ; extract speed
00F5 C5 F607 CF43 90 0A89 1741 MOVB YF$DHU_SPEEDS[R3],UCBSW_TT_SPEED+1(R5); convert to VMS value
0A91 1742
0A91 1743 : insert other parameters in the new LPR value
0A91 1744
0A91 1745 20$:
0A91 1746
53 00F8 C5 02 03 EE 0A91 1747 EXTV #UCBSV_TT_LEN,#2,UCBSB_TT_PARITY(R5),R3 ; GET CHAR SIZE
6E 02 03 53 F0 0A98 1748 INSV R3,#DHULPR$V_SIZE,#2,(SP) ; SET IT
53 00F8 C5 02 06 EE 0A9D 1750 EXTV #UCBSV_TT_PARTY,#2,UCBSB_TT_PARITY(R5),R3 ; GET PARITY/ODD
6E 02 05 53 F0 0AA4 1751 INSV R3,#DHULPR$V_PARITY,#2,(SP)
6E 00000040 8F CC 0AA9 1752 XORL #DHULPR$M_ODD,(SP) ; Correct sense
0A80 1753
53 00F8 C5 01 05 EE 0AB0 1754 EXTV #UCBSV_TT_STOP,#1,UCBSB_TT_PARITY(R5),R3 ; GET STOP
6E 01 07 53 F0 0AB7 1755 INSV R3,#DHULPR$V_STOP,#1,(SP)
0ABC 1756
04 A0 6E B1 0ABC 1757 CMPW (SP),DHULPR(R0) ; Any modifications to be made?
04 13 0AC0 1758 BEQL 30$ ; EQL, no then return
04 A0 6E F7 0AC2 1759 CRTLW (SP),DHULPR(R0) ; INSERT AS LINE PARAMETER
5E 04 6E C0 0AC6 1760 ADDL #4,SP ; Restore stack
OC BA 0AC9 1761 POPR #M<R2,R3> ; Restore registers
05 0ACB 1762 RSB
0ACC 1763
0ACC 1764
0ACC 1765
0ACC 1766 YF$END: ; end of driver
```

YFDRIVER
V04-000

- Port Driver for DHU/DHV
SET SPEED, PARITY PARAMETERS

OACC 1767 .END

M 8

16-SEP-1984 02:26:48 VAX/VMS Macro V04-00
5-SEP-1984 04:17:43 [TTDRVR.SRC]YFDRIVER.MAR;1

Page 41
(1)

-\$25

Symb

FOUN

GENE

GETC

GETC

GETC

GETC

GETC

GETC

GETC

GETC

GETC

GETC

GETC

GRP

HELP

HOLD

IDEN

LBR

LETT

LGIS

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

LIB

YFDRIVER
Symbol table

- Port Driver for DHU/DHV

N 8

16-SEP-1984 02:26:48 VAX/VMS Macro V04-00
5-SEP-1984 04:17:43 [TTDRVR.SRC]YFDRIVER.MAR;1

Page 42
(1)

```

$$$ = 00000020 R 02
$$OP = 00000002
ATS_UBA = 00000001
BIT... = 00000003
BUG$ UNSUPRTCPU *****
BURST_OUTPUT 00000443 R X 03
CLASS-DDT = 00000010
CLASS-DS_TRAN = 0000000C
CLASS-FORK = 0000001C
CLASS-GETNXT = 00000000
CLASS-POWERFAIL = 00000020
CLASS-PUTNXT = 00000004
CLASS-READERROR = 00000014
CLASS-SETUP_UCB = 00000008
CRBSB-TT TYPE = 0000000B
CRBSL-INTD = 00000024
CRBSL-INTD2 = 00000048
DC$ TERM = 00000042
DDBSL-DDT = 0000000C
DEVSM-AVL = 00040000
DEVSM-CCL = 00000002
DEVSM-IDV = 04000000
DEVSM-NNM = 00000200
DEVSM-ODV = 08000000
DEVSM-REC = 00000001
DEVSM-TRM = 00000004
DHUCSR = 00000000
DHUCSR$C_BASE = 00004040
DHUCSR$M_CLEAR = 00000020
DHUCSR$M-DIGFAL = 00002000
DHUCSR$M-DMAERR = 00001000
DHUCSR$M-IADDR = 0000000F
DHUCSR$M-LINE = 00000F00
DHUCSR$M-RCVINT = 00000040
DHUCSR$M-SNDINT = 00004000
DHUCSR$M-SNDRDY = 00008000
DHUCSR$S_CLEAR = 00000001
DHUCSR$S-DIGFAL = 00000001
DHUCSR$S-DMAERR = 00000001
DHUCSR$S-IADDR = 00000004
DHUCSR$S-LINE = 00000004
DHUCSR$S-RCVINT = 00000001
DHUCSR$S-SNDINT = 00000001
DHUCSR$S-SNDRDY = 00000001
DHUCSR$V_CLEAR = 00000005
DHUCSR$V-DIGFAL = 0000000D
DHUCSR$V-DMAERR = 0000000C
DHUCSR$V-IADDR = 00000000
DHUCSR$V-LINE = 00000008
DHUCSR$V-RCVINT = 00000006
DHUCSR$V-SNDINT = 0000000E
DHUCSR$V-SNDRDY = 0000000F
DHULCT = 00000008
DHULCT$M_ABORT = 00000001
DHULCT$M-BREAK = 00000008
DHULCT$M-DTR = 00000200
DHULCT$M-IAUTO = 00000010

```

```

DHULCT$M_MAINT = 000000C0
DHULCT$M-MODEM = 00000100
DHULCT$M-OAUTO = 00000002
DHULCT$M-RCV = 00000004
DHULCT$M-RTS = 00001000
DHULCT$M-SNDOFF = 00000020
DHULCT$S_ABORT = 00000001
DHULCT$S-BREAK = 00000001
DHULCT$S-DTR = 00000001
DHULCT$S-IAUTO = 00000001
DHULCT$S-MAINT = 00000002
DHULCT$S-MODEM = 00000001
DHULCT$S-OAUTO = 00000001
DHULCT$S-RCV = 00000001
DHULCT$S-RTS = 00000001
DHULCT$S-SNDOFF = 00000001
DHULCT$V_ABORT = 00000000
DHULCT$V-BREAK = 00000003
DHULCT$V-DTR = 00000009
DHULCT$V-IAUTO = 00000004
DHULCT$V-MAINT = 00000006
DHULCT$V-MODEM = 00000008
DHULCT$V-OAUTO = 00000001
DHULCT$V-RCV = 00000002
DHULCT$V-RTS = 0000000C
DHULCT$V-SNDOFF = 00000005
DHULPR = 00000004
DHULPR$M_DIAG1 = 00000002
DHULPR$M_DIAG2 = 00000004
DHULPR$M-ODD = 00000040
DHULPR$M-PARITY = 00000020
DHULPR$M-RSPEED = 00000F00
DHULPR$M-SIZE = 00000018
DHULPR$M-STOP = 00000080
DHULPR$M-TSPEED = 0000F000
DHULPR$S_DIAG1 = 00000001
DHULPR$S_DIAG2 = 00000001
DHULPR$S-ODD = 00000001
DHULPR$S-PARITY = 00000001
DHULPR$S-RSPEED = 00000004
DHULPR$S-SIZE = 00000002
DHULPR$S-STOP = 00000001
DHULPR$S-TSPEED = 00000004
DHULPR$V_DIAG1 = 00000001
DHULPR$V_DIAG2 = 00000002
DHULPR$V-ODD = 00000006
DHULPR$V-PARITY = 00000005
DHULPR$V-RSPEED = 00000008
DHULPR$V-SIZE = 00000003
DHULPR$V-STOP = 00000007
DHULPR$V-TSPEED = 0000000C
DHURBF = 00000002
DHURCV$M_BUF = 000000FF
DHURCV$M-FRAMER = 00002000
DHURCV$M-LINE = 00000F00
DHURCV$M-OVERRUN = 00004000
DHURCV$M-PARERR = 00001000

```

_\$2

Sym

PAR
PAR
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PRO
PRO
PRV
PRV
PRV
PRV
PRV
PUR
PWD
R
RAB
RAN
RAN
RDB
RDB
REC
REC
REC
REM
REN
RES
RES
RIG
RMS
SET
SET
SHO
SHO
SIG
SS\$
SS\$
SS\$
SS\$
STR
STR

YFDRIVER
Symbol table

- Port Driver for DHU/DHV

B 9

16-SEP-1984 02:26:48 VAX/VMS Macro V04-00
5-SEP-1984 04:17:43 [TTDRVR.SRC]YFDRIVER.MAR;1Page 43
(1)

DHURCVSM_VALID	=	00008000		
DHURCVSS_BUF	=	00000008		
DHURCVSS_FRAMER	=	00000001		
DHURCVSS_LINE	=	00000004		
DHURCVSS_OVERRUN	=	00000001		
DHURCVSS_PARERR	=	00000001		
DHURCVSS_VALID	=	00000001		
DHURCVSV_BUF	=	00000000		
DHURCVSV_FRAMER	=	0000000D		
DHURCVSV_LINE	=	00000008		
DHURCVSV_OVERRUN	=	0000000E		
DHURCVSV_PARERR	=	0000000C		
DHURCVSV_VALID	=	0000000F		
DHUSPDSC_BAUD_110	=	00000002		
DHUSPDSC_BAUD_1200	=	00000007		
DHUSPDSC_BAUD_134	=	00000003		
DHUSPDSC_BAUD_150	=	00000004		
DHUSPDSC_BAUD_1800	=	00000008		
DHUSPDSC_BAUD_19200	=	0000000E		
DHUSPDSC_BAUD_2000	=	00000009		
DHUSPDSC_BAUD_2400	=	0000000A		
DHUSPDSC_BAUD_300	=	00000005		
DHUSPDSC_BAUD_38400	=	0000000F		
DHUSPDSC_BAUD_4800	=	0000000B		
DHUSPDSC_BAUD_50	=	00000000		
DHUSPDSC_BAUD_600	=	00000006		
DHUSPDSC_BAUD_7200	=	0000000C		
DHUSPDSC_BAUD_75	=	0000C001		
DHUSPDSC_BAUD_9600	=	0000000D		
DHUSTT	=	00000007		
DHUSTTSM_CTS	=	00000008		
DHUSTTSM_DCD	=	00000010		
DHUSTTSM_DSR	=	00000080		
DHUSTTSM_RI	=	00000020		
DHUSTTSS_CTS	=	00000001		
DHUSTTSS_DCD	=	00000001		
DHUSTTSS_DSR	=	00000001		
DHUSTTSS_RI	=	00000001		
DHUSTTSV_CTS	=	00000003		
DHUSTTSV_DCD	=	00000004		
DHUSTTSV_DSR	=	00000007		
DHUSTTSV_RI	=	00000005		
DHUTBF1	=	0000000A		
DHUTBF2	=	0000000C		
DHUTCR	=	00000002		
DHUTCT	=	0000000E		
DHUTFS	=	00000006		
DHUTXC	=	00000002		
DHUTXF	=	00000006		
DMA_ALLOC	00000522	R	03	
DMA_CONTINUE	0000055D	R	03	
DMA_DEALLOC	000006E8	R	03	
DMA_DONE	000006D4	R	03	
DMA_MAP	0000061D	R	03	
DMA_NOMAP	0000057D	R	03	
DMA_POST	00000708	R	03	
DMA_START	000004F1	R	03	

DPTSC_LENGTH	=	00000038		
DPTSC_VERSION	=	00000004		
DPT\$INITAB	=	00000038	R	02
DPTSM_NOUNLOAD	=	00000004		
DPT\$REINITAB	=	000000D3	R	02
DPT\$TAB	=	000000C0	R	02
DPT\$W_VECTOR	=	0000001E		
DT\$_DHU	=	00000047		
DT\$_DHV	=	00000046		
DYN\$C_CRB	=	00000005		
DYN\$C_DDB	=	00000006		
DYN\$C_DPT	=	0000001E		
DYN\$C_ORB	=	00000049		
DYN\$C_UCB	=	00000010		
EXESGB_CPUTYPE	*****	X	03	
EXESGL_ABSTIM	*****	X	03	
EXESGL_TENUSEC	*****	X	03	
EXESGL_UBDELAY	*****	X	03	
FUNCTAB_LEN	=	00000000		
IDBSL_UCBLST	=	00000018		
INIT_CONTINUE	=	00000235	R	03
IOSM_AUTXOF_DIS	=	00004000		
IOSM_AUTXOF_ENA	=	00002000		
IOSM_LINE_OFF	=	00000200		
IOSM_LINE_ON	=	00000800		
IOSM_LOOP	=	00000080		
IOSM_UNLOOP	=	00000100		
IOCSALOUBAMAPN	*****	X	03	
IOCSMNTVER	*****	X	03	
IOCSRELMAPREG	*****	X	03	
IOCSRETURN	*****	X	03	
MMGSGL_SPTBASE	*****	X	03	
MODEM\$C_DATASET	=	00000003		
MODEM\$C_INIT	=	00000000		
ORBSB_FLAGS	=	0000000B		
ORBSL_OWNER	=	00000000		
ORBSM_PROT_16	=	00000001		
ORBSW_PROT	=	00000018		
PORT_ABORT	=	00000020		
PORT_DS_SET	=	0000000C		
PORT_FORKRET	=	00000034		
PORT_LENGTH	=	00000038		
PORT_MAINT	=	00000030		
PORT_RESUME	=	00000024		
PORT_SET_LINE	=	00000008		
PORT_STARTIO	=	00000000		
PORT_STOP	=	00000018		
PORT_VECTOR	=	00000048	R	03
PORT_XOFF	=	00000014		
PORT_XON	=	00000010		
PR\$_IPL	=	00000012		
PR\$_SID_TYP780	=	00000001		
SILD_OUTPUT	=	00000469	R	03
SIZ...	=	00000001		
SS\$_NORMAL	=	00000001		
TT\$C_BAUD_110	=	00000003		
TT\$C_BAUD_1200	=	00000008		


```

= 0C000124
= 00000125
= 0000012A
= 000000F8
= 0000010A
= 00000134
= 00000024
= 00000028
= 00000088
= 0000003C
= 0000003C
= 00000044
= 00000048
= 0000006C
= 0000012C
= 00000114
= 000000C8
= 000000C4
= 0000010C
= 0000011C
= 00000118
= 00000110
= 000000B4
= 000000D0
= 000000CC
= 00000002
= 00000002
= 00000001
= 00000010
= 00000001
= 00000004
= 00000001
= 00000001
= 00000001
= 00000001
= 00000001
= 00000001
= 00000000
= 00000004
= 00000005
= 00000007
= 00000003
= 00000006
= 00000005
= 00000002
= 00000042
= 00000064
= 000000F1
= 000000E8
= 00000108
= 00000120
= 00000122
= 000000F4
= 00000106
= 00000054
= 00000014
= 00000008
= 0000000C

```

[illegible]

VECSL_UNITINIT	=	00000018		
VECSS_MAPREG	=	0000000F		
VECSV_MAPREG	=	00000000		
VECSW_MAPREG	=	00000010		
YFSABORT		000007E4	R	03
YFSCTRL_ERROR		00000155	R	03
YFSDDT		00000000	RG	03
YFSDELIVER		000000A4	RG	03
YFSDHU_SPEEDS		00000095	R	03
YFSDPT		00000000	RG	02
YFSDS_SET		0000033C	R	03
YFSEND		00000ACC	R	03
YFSFORK		0000071F	R	03
YFSINITIAL		000000C8	RG	03
YFSINITLINE		00000156	RG	03
YFSINTINP		0000036F	RG	03
YFSINTOUT		00000897	RG	03
YFSL_DMAXMT_ERROR		00000040	RG	03
YFSL_ERROR		0000003C	RG	03
YFSL_INACT ERROR		00000044	RG	03
YFSL_SIL_ERROR		00000038	RG	03
YFSMAINT		0000029B	R	03
YFSNULL		00000084	R	03
YFSPREEMPT		00000777	R	03
YFSRESUME		00000806	R	03
YFSSET_LINE		000009EB	R	03
YFSSTARTIO		0000043E	RG	03
YFSSTOP		000007C4	R	03
YFSUNIT_ERROR		00000296	R	03
YFSVEC		00000048	R	03
YFSVECEND		00000080	R	03
YFSVMS SPEEDS		00000085	R	03
YFSXOFF		0000075E	R	03
YFSXON		0000073E	R	03
YF_DMA_INTERRUPT		000009B1	R	03
YF_INITMAP		0000022E	R	03
YF_INITNULL		00000227	R	03
YF_NOMAP		00000235	R	03
YF_OUT_EXIT		0000088A	R	03
YF_OUT_LOOP		00000897	R	03
YF_PREEMPT		0000098F	R	03
YF_SILO		00000910	R	03
YF_START BURST		00000909	R	03

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes											
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE	
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE	
\$\$\$105_PROLOGUE	000000E8 (232.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE	
\$\$\$115_DRIVER	00000ACC (2764.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG	

—\$2

[illegible]

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
-----	-----	-----	-----
Initialization	29	00:00:00.04	00:00:00.19
Command processing	123	00:00:00.39	00:00:01.17
Pass 1	754	00:00:23.82	00:00:27.56
Symbol table sort	0	00:00:03.37	00:00:03.47
Pass 2	296	00:00:04.93	00:00:05.45
Symbol table output	44	00:00:00.23	00:00:00.45
Psect synopsis output	2	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1250	00:00:32.79	00:00:38.30

The working set limit was 2550 pages.

198247 bytes (388 pages) of virtual memory were used to buffer the intermediate code.

There were 170 pages of symbol table space allocated to hold 3101 non-local and 128 local symbols.

1767 source lines were read in Pass 1, producing 24 object records in Pass 2.

80 pages of virtual memory were used to define 74 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	35
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	47

3572 GETS were required to define 47 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:YFDRIVER/OBJ=OBJ\$:YFDRIVER MSRC\$:YFDRIVER/UPDATE=(ENH\$:YFDRIVER)+EXECMLS/LIB

0405 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

